

MIT OpenCourseWare
<http://ocw.mit.edu>

MAS.632 Conversational Computer Systems
Fall 2008

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.

3

Speech Coding

This chapter introduces the methods of encoding speech digitally for use in such diverse environments as talking toys, compact audio discs, and transmission over the telephone network. To use voice as data in such diverse applications as voice mail, annotation of a text document, or a talking telephone book, it is necessary to store speech in a form that enables computer retrieval. Digital representations of speech also provide the basis for both speech synthesis and voice recognition. Chapter 4 discusses the management of stored voice for computer applications, and Chapter 6 explore interface issues for interactive voice response systems.

Although the use of a conventional analog tape recorder for voice storage is an option in consumer products such as answering machines, its use proves impractical in more interactive applications. Tape is serial, requiring repeated use of fast-forward and rewind controls to access the data. The tape recorder and the tape itself are prone to mechanical breakdown; standard audiotape is unable to endure the cutting and splicing of editing.

Alternatively, various digital coding schemes enable a sound to be manipulated in computer memory and stored on random access magnetic disks. Speech coding algorithms provide a number of digital representations of a segment of speech. The goal of these algorithms is to faithfully capture the voice signal while utilizing as few bits as possible. Minimizing the data rate conserves disk storage space and reduces the bandwidth required to transmit speech over a network; both storage and bandwidth may be scarce resources and hence costly. A wide range of speech quality is available using different encoding algorithms; quality requirements vary from application to application. In general, speech quality may be traded for information rate by choosing an appropriate coding algorithm.

Some formerly popular coding schemes, such as Linear Predictive Coding, made sizable sacrifices of quality for a low bit rate. While this may have been dictated by the technology of the time, continually decreasing storage costs and increasingly available network bandwidth now allow better-quality speech coding for workstation environments. The poorer-quality encoders do find use, however, in military environments where encryption or low transmitter power may be desired and in consumer applications such as toys where reasonable costs are essential.

The central issue in this chapter is the proposed trade-off between bit rate and intelligibility; each generation of speech coder attempts to improve speech quality and decrease data rate. However, speech coders may incur costs other than limited intelligibility. Coder complexity may be a factor; many coding algorithms require specialized hardware or digital signal processing microprocessors to run in real time.¹ Additionally, if the coder uses knowledge of the history of the signal to reduce the data rate, some flexibility may be lost in the encoding scheme, making it difficult both to edit the stored speech and to evaluate the encoded signal at a particular moment in time. And finally, although it is of less concern here, coder robustness may be an issue; some of the algorithms fail less gracefully, with increased sound degradation if the input data is corrupted.

After discussing the underlying process of digital sampling of speech, this chapter introduces some of the more popular speech coding algorithms. These algorithms may take advantage of a priori knowledge about the characteristics of the speech signal to encode only the most perceptually significant portions of the signal, reducing the data rate. These algorithms fall into two general classes: **waveform coders**, which take advantage of knowledge of the speech signal itself and **source coders**, which model the signal in terms of the vocal tract.

There are many more speech coding schemes than can be detailed in a single chapter of this book. The reader is referred to [Flanagan et al. 1979, Rabiner and Schafer 1978] for further discussion. The encoding schemes mentioned here are currently among the most common in actual use.

SAMPLING AND QUANTIZATION

Speech is a continuous, time-varying signal. As it radiates from the head and travels through the air, this signal consists of variations in air pressure. A microphone is used as a transducer to convert this variation in air pressure to a corresponding variation in voltage in an electrical circuit. The resulting electrical signal is analog. An analog signal varies smoothly over time and at any moment is one of an unlimited number of possible values within the dynamic range of the

¹A real-time algorithm can encode and decode speech as quickly as it is produced. A non-real-time algorithm cannot keep up with the input signal; it may require many seconds to encode each second of speech.

medium in which the signal is transmitted.² In analog form, the signal may be transmitted over a telephone circuit (as voltage) or stored on a magnetic tape (as magnetic flux).

All the sensory stimuli we encounter in the physical world are analog. For an analog signal to be stored or processed by a computer, it must first be transformed into a digital signal, i.e., **digitized**. Digitization is the initial step of signal processing for speech recognition. Speech synthesis algorithms also generate digital signals, which are then transformed into analog form so we can hear them.

A digitized signal consists of a series of numeric values at some regular rate (the sampling rate). The signal's possible values are constrained by the number of bits available to represent each sampled value—this simple representation is called **Pulse Code Modulation (PCM)**. To convert an analog signal to this digital form it must be **sampled** and **quantized**.

Sampling assigns a numeric value to the signal in discrete units of time, i.e., a new value is given for the signal at a constant rate. The Nyquist rate specifies the maximum frequency of the signal that may be fully reproduced: a signal must be sampled at twice its highest frequency to be faithfully captured [Nyquist 1924]. In addition, when the analog signal is being recreated, it must be passed through a low-pass filter at one-half the sampling rate to remove audible artifacts of the sampling frequency.³

If these sampling rate conditions are not met, **aliasing** results. Aliasing is a form of distortion in which the high frequencies are falsely regenerated as lower frequencies. A rigorous description of aliasing requires mathematics beyond the scope of this book; nonetheless, the following example offers a degree of explanation. While watching a western film, you may have noticed that the wagon wheels sometimes appear to rotate backwards or in the correct direction but at the wrong rate; the same phenomenon occurs with airplane propellers, fans, etc. This is an example of aliasing. In actuality, the wheel is turning continuously, but film captures images at a rate of 24 frames per second. A repetitive motion that occurs at more than half this sampling rate cannot be captured correctly; when we view the film we perceive a motion that may be faster, slower, or of opposite direction.

Why does this happen? Consider the eight-spoked wagon wheel shown in Figure 3.1; as it rotates clockwise, the spokes move together. If the spokes move just a little in the $\frac{1}{4}$ of a second between frames (see Figure 3.2), they will appear to be moving forward at the correct speed. If the wheel moves exactly $\frac{1}{2}$ of a revolution between frames, the spokes will have changed position (see Figure 3.3); but since the spokes all look identical, the film audience will not see the spokes rotating even though they have moved forward. This is, of course, an illusion, as the stagecoach would be moving against the background scenery. If the wheel rotates slightly faster such that each spoke has traveled beyond the position originally

²An electric circuit saturates at a maximum voltage, and below a minimum level the voltage variations are not detectable.

³In practice, to ensure that the first condition is actually met during sampling, a similar filter at one half the sampling frequency is used at the input stage.

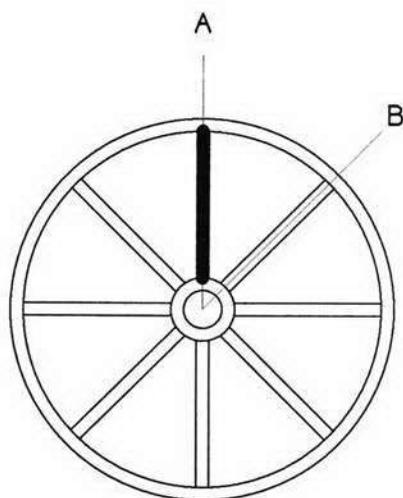


Figure 3.1. A wagon wheel with two spokes marked for reference points.

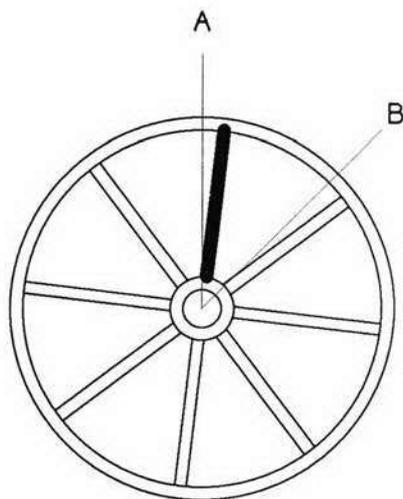


Figure 3.2. The wheel has rotated forward a small distance.

occupied by the next spoke (see Figure 3.4), the viewer will see a much slower motion identical to that depicted by Figure 3.2. Finally, if a spoke has moved most of the way to the next spoke position between frames (see Figure 3.5), it will appear that the wheel is slowly moving counterclockwise.

Except for the first case, these illusions are all examples of aliasing. The repetitive motion consists of identical spokes rotating into positions occupied by other

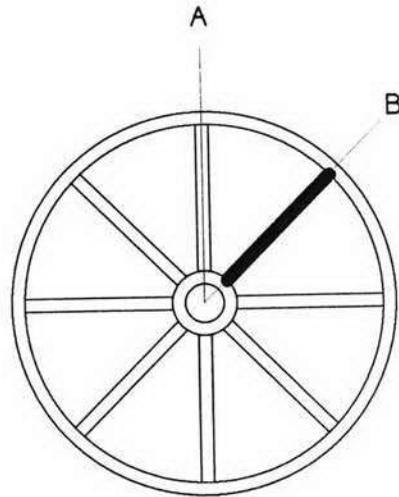


Figure 3.3. The wheel has rotated far enough that a spoke has moved forward to the position previously occupied by the next spoke.

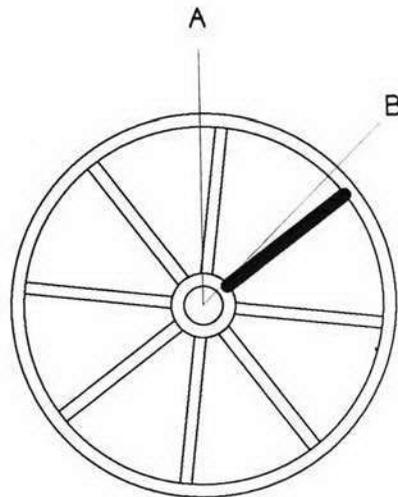


Figure 3.4. The wheel has rotated still further. Each spoke has moved just beyond the position formerly occupied by the next.

spokes in the time between frames. If aliasing is to be avoided, the sampling theorem suggests that this must happen at no more than one-half the sampling rate. This implies that each spoke must rotate no more than one-half the distance to the next spoke between frames; with eight spokes filmed at 24 frames per second, this corresponds to $1\frac{1}{2}$ wheel revolutions per second. A characteristic

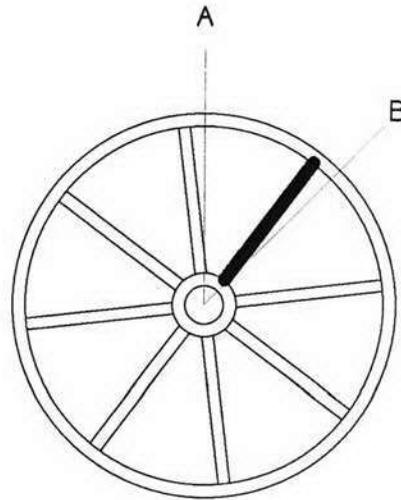


Figure 3.5. The spokes have moved almost to the position of the next spoke.

of aliasing is that frequencies above one-half the sampling rate appear incorrectly as lower frequencies. Identical aliasing artifacts also appear at multiples of the sampling frequency. The visual effects of the wheel moving a small fraction of the way to the next spoke, the distance to the next spoke plus the same small fraction, or the distance to the second spoke beyond plus the small fraction are all identical.

Another example provides a better understanding of sampling applied to an audio signal. Consider a relatively slowly varying signal as shown in Figure 3.6. If sampled at some rate T , a series of values that adequately capture the original

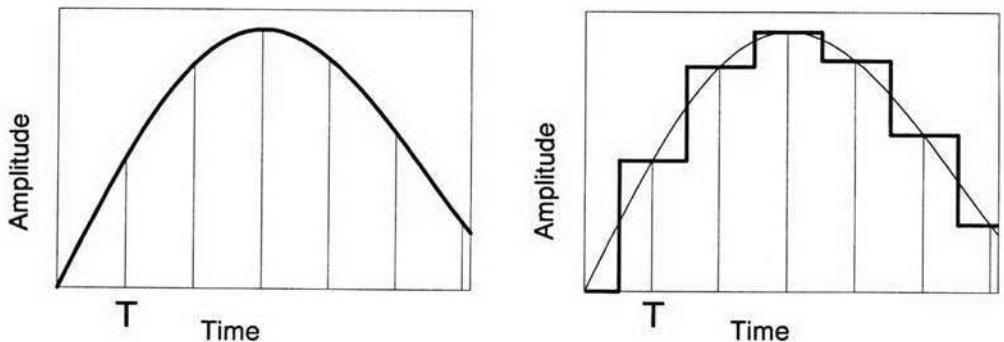


Figure 3.6. Sampling and reconstruction of a slowly varying signal. The continuously varying signal on the left is sampled at discrete time intervals T . The reconstructed signal is shown overlaid on the original in the right figure.

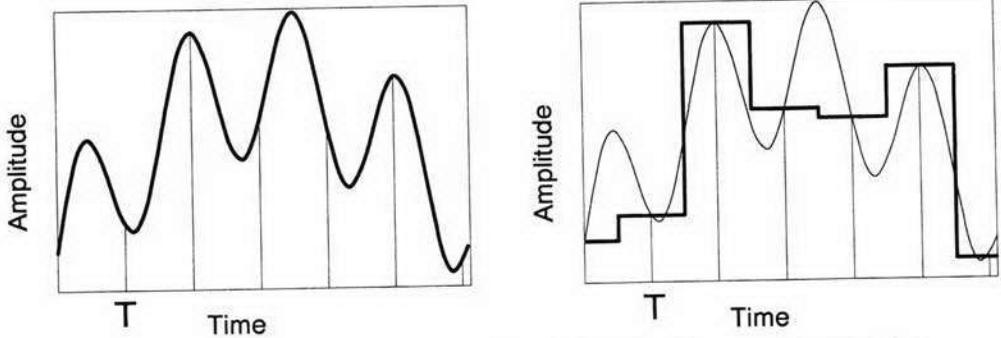


Figure 3.7. Inadequate reconstruction of a higher frequency signal. The original signal varies more rapidly than that in Figure 3.6, and the resulting reconstruction on the right shows gross errors.

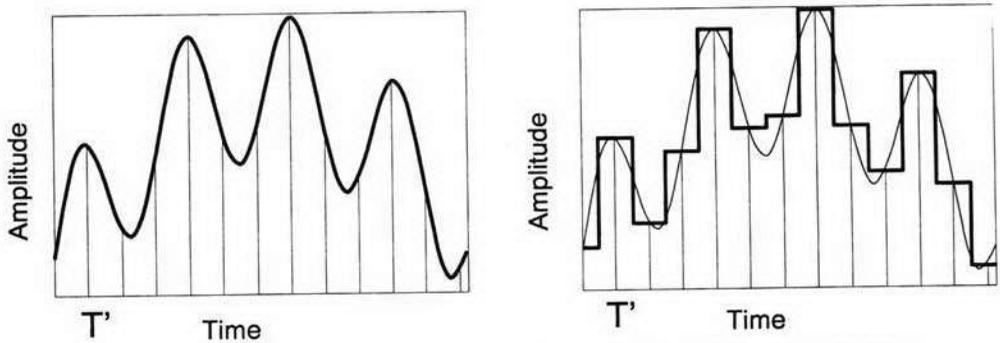


Figure 3.8. The higher frequency signal sampled at a higher rate, T' . The reconstructed signal now more accurately tracks the original.

signal are obtained. If the signal varies more rapidly (i.e., has components of a higher frequency), the same sampling rate may not correctly capture the signal (see Figure 3.7). Instead a higher sampling rate (smaller T') is required to capture the more rapidly changing signal (see Figure 3.8); this signal has a higher **bandwidth**.

Bandwidth is the range of frequencies contained in a signal. For speech these span from a low frequency at the fundamental frequency of voicing (perhaps 100 to 400 Hertz—lower for men, higher for women, and higher still for children) to upwards of 15 kHz. The telephone network supports a voice bandwidth of approximately 3100 Hz (see Chapter 10 for more details); speech is sampled at 8000 Hz for digital telephone transmission.⁴ Although this bandwidth is adequate for intelligibility, the transmitted voice is clearly of lower quality than face-to-face

⁴If ideal filters really existed, this sampling rate could be reduced to 6200Hz (twice 3100 Hz), but real filters have a gradual frequency cutoff as opposed to the discontinuous response of ideal filters.

communication. Compare this with a compact audio disc, which samples at 44.1 kHz giving a frequency response in the range of 20 kHz. Although the higher bandwidth is required more for the musical instruments than for the singers, the difference in voice quality offered by the increased bandwidth response is striking as well. Of course the compact disc requires a sampling rate five times greater than the telephone with the resulting increased requirements for storage and transmission rate.

Sampling rate is only one consideration of digitization; the other is the resolution of the samples at whatever rate they are taken. Each sample is represented as a digital value, and that number covers only a limited range of discrete values with reference to the original signal. For example, if 4 bits are allowed per sample, the sample may be one of 16 values; while if 8 bits are allowed per sample, it may be one of 256 values. The size in terms of number of bits of the samples can be thought of as specifying the fineness of the spacing between the lines in Figure 3.9. The closer the spacing between possible sample values, the higher the resolution of the samples and the more accurately the signal can be reproduced. In Figure 3.9, the **error**, or difference between the original and reconstructed signals, is shaded; higher resolution results in decreased quantization error.

The resolution of the quantizer (the number of bits per sample) is usually described in terms of a **signal-to-noise ratio**, or **SNR**. The error between the quantized sample and the original value of the signal can be thought of as noise, i.e., the presence of an unwanted difference signal that is not correlated with the original. The higher the SNR, the greater the fidelity of the digitized signal. SNR for PCM encoding is proportional to two to the power of the number of bits per sample, i.e., $\text{SNR} \approx 2^B$ where B is bits per sample. Adding one bit to the quantizer doubles the SNR.

Note that the SNR, or quantization error, is independent of the sampling rate limitation on bandwidth. The data rate of a sampled signal is the product of the sample size times the sampling rate. Increasing either or both increases

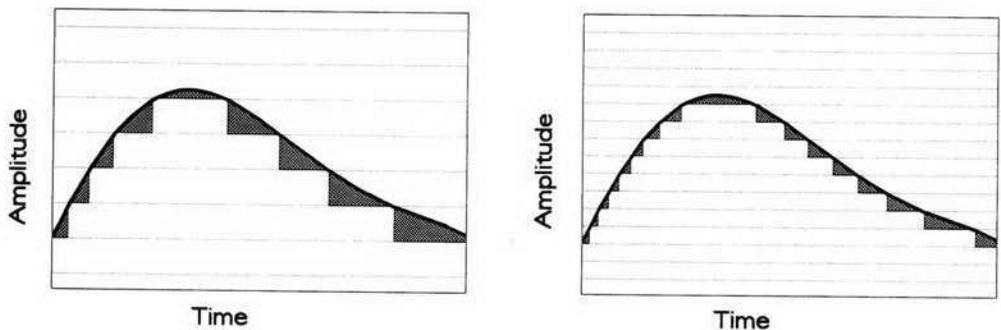


Figure 3.9. Low and high resolution quantization. Even when the signal is sampled continuously in time, its amplitude is quantized, resulting in an error, shown by the shaded regions.

the data rate. But increasing the sampling rate has no effect on quantization error and vice versa.

To compare the telephone and the compact audio disc again, consider that the telephone uses 8 bits per sample, while the compact disc uses 16 bits per sample. The telephone samples at 8 kHz (i.e., 8000 samples per second), while the compact disc sampling rate is 44.1 kHz. One second of telephone-quality speech requires 64,000 bits, while one second of compact disc-quality audio requires an order of magnitude more data, about 700,000 bits, for a single channel.⁵

Quantization and sampling are usually done with electronic components called **analog-to-digital** (A/D) and **digital-to-analog** (D/A) converters. Since one of each is required for bidirectional sampling and reconstruction, they are often combined in a **codec** (short for coder/decoder), which is implemented as a single integrated circuit. Codecs exist in all digital telephones and are being included as standard equipment in a growing number of computer workstations.

SPEECH CODING ALGORITHMS

The issues of sampling and quantization apply to any time-varying digital signal. As applied to audio, the signal can be any sound: speech, a cat's meow, a piano, or the surf crashing on a beach. If we know the signal is speech, it is possible to take advantage of characteristics specific to voice to build a more efficient representation of the signal. Knowledge about characteristics of speech can be used to make limiting assumptions for waveform coders or models of the vocal tract for source coders.

This section discusses various algorithms for **coders** and their corresponding **decoders**. The coder produces a digital representation of the analog signal for either storage or transmission. The decoder takes a stream of this digital data and converts it back to an analog signal. This relationship is shown in Figure 3.10. Note that for simple PCM, the coder is an A/D converter with a clock to generate a sampling frequency and the decoder is a D/A converter. The decoder can derive a clock from the continuous stream of samples generated by the coder or have its own clock; this is necessary if the data is to be played back later from storage. If the decoder has its own clock, it must operate at the same frequency as that of the coder or the signal is distorted.⁶

Waveform Coders

An important characteristic of the speech signal is its dynamic range, the difference between the loudest and most quiet speech sounds. Although speech has a

⁵The compact disc is stereo; each channel is encoded separately, doubling the data rate given above.

⁶In fact it is impossible to guarantee that two independent clocks will operate at precisely the same rate. This is known as clock skew. If audio data is arriving in real time over a packet network, from time to time the decoder must compensate by skipping or inserting data.

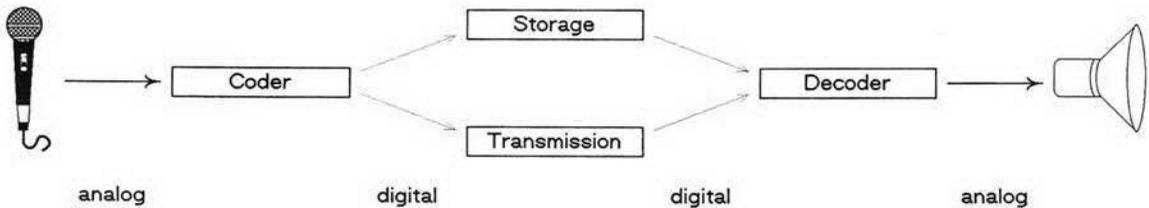


Figure 3.10. The coder and decoder are complementary in function.

fairly wide dynamic range, it reaches the peak of this range infrequently in normal speech. The amplitude of a speech signal is usually significantly lower than it is at occasional peaks, suggesting a quantizer utilizing a nonlinear mapping between quantized value (output) and sampled value (input). For very low signal values the difference between neighboring quantizer values is very small, giving fine quantization resolution; however, the difference between neighbors at higher signal values is larger, so the coarse quantization captures a large dynamic range without using more bits per sample. The mapping with nonlinear quantization is usually logarithmic. Figure 3.11 illustrates the increased dynamic range that is achieved by the logarithmic quantizer.

Nonlinear quantization maximizes the information content of the coding scheme by attempting to get roughly equal use of each quantizer value averaged over time. With linear coding, the high quantizer values are used less often; logarithmic coding compensates with a larger step size and hence coarser resolution at the high extreme. Notice how much more of the quantizer range is used for the lower amplitude portions of the signal shown in Figure 3.12. Similar effects can also be accomplished with adaptive quantizers, which change the step size over time.

A piece-wise linear approximation of logarithmic encoding, called μ -law,⁷ at 8 bits per sample is standard for North American telephony.⁸ In terms of signal-to-noise ratio, this 8 bits per sample logarithmic quantizer is roughly equivalent to 12 bits per sample of linear encoding. This gives approximately 72 dB of dynamic range instead of 48 dB, representing seven orders of magnitude of difference between the highest and lowest signal values that the coder can represent.

Another characteristic of speech is that adjacent samples are highly correlated; neighboring samples tend to be similar in value. While a speech signal contains frequency components up to one-half the sampling rate, the higher-frequency components are of lower energy than the lower-frequency components. Although the signal can change rapidly, the amplitude of the rapid changes is less than that of slower changes. This means that differences between adjacent samples are less than the overall range of values supported by the quantizer so fewer bits are required to represent differences than to represent the actual sample values.

⁷Pronounced "mu-law."

⁸A very similar scale, "A-law," is used in Europe.

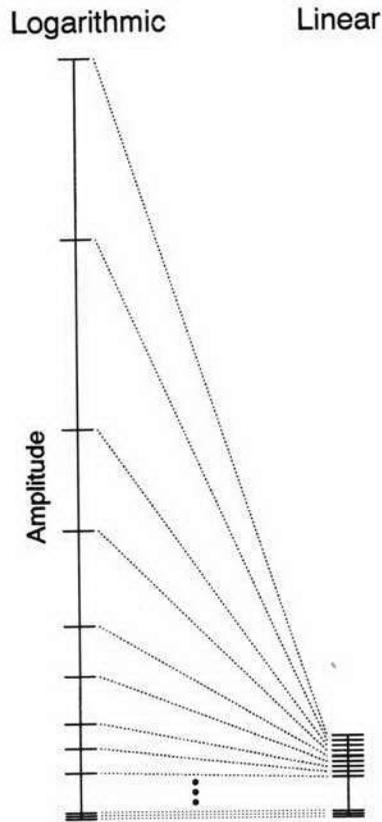


Figure 3.11. A comparison of dynamic range of logarithmic and linear coders for the same number of bits per sample. The logarithmic coder has a much greater dynamic range than a linear coder with comparable resolution at low amplitudes.

Encoding of the differences between samples is called **Delta PCM**. At each sampling interval the encoder produces the difference between the previous and the current sample value. The decoder simply adds this value to the previous value; the previous value is the only information it needs to retain. This process is illustrated in Figure 3.13.

The extreme implementation of difference coding is **Delta Modulation**, which uses just one bit per sample. When the signal is greater than the last sample, a "1" bit results; when less, a "0" bit results. During reconstruction, a constant step size is added to the current value of the signal for each "1" and subtracted for each "0."

This simple coder has two error modes. If the signal increases or decreases more rapidly than the step size times the sampling rate, **slope overload** results as the coder cannot keep up. This may be seen in the middle of Figure 3.14. To

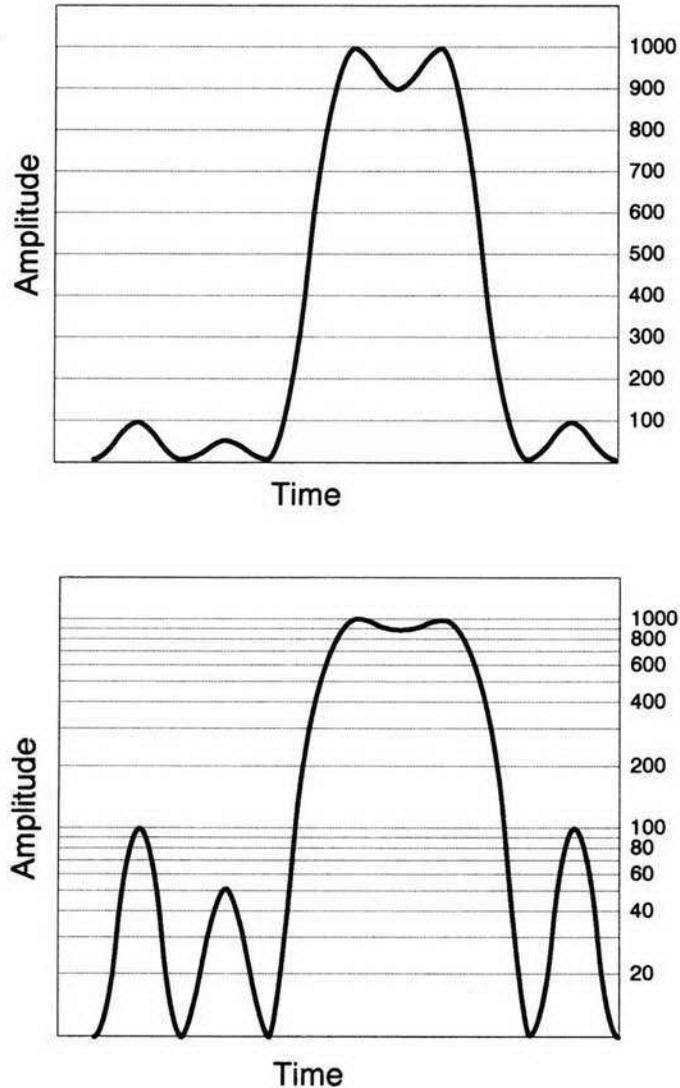


Figure 3.12. Use of a logarithmic scale disperses the signal density over the range of the coder. The top figure shows a linearly encoded signal; the bottom shows the same signal logarithmically encoded.

compensate for this effect, sampling rates significantly in excess of the Nyquist rate (e.g., in the range of 16 to 32 kHz) are usually used. The second error mode occurs when the delta-modulated signal has a low value such as in moments of silence. The reconstructed signal must still alternate up and down (see the right side of Figure 3.14) as the 1-bit coder cannot indicate a “no-change” condition. This is called **quieting noise**.

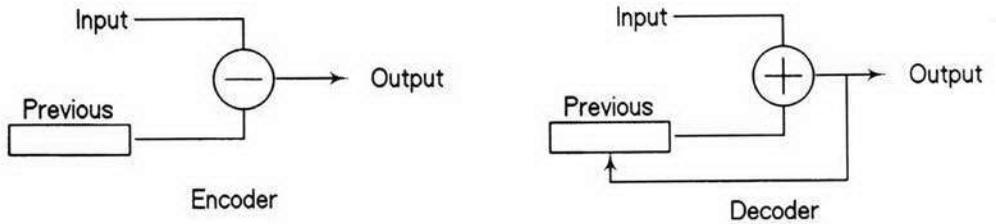


Figure 3.13. The Delta PCM encoder transmits the difference between sample values. The signal is reconstructed by the decoder, which always stores the previous output value.

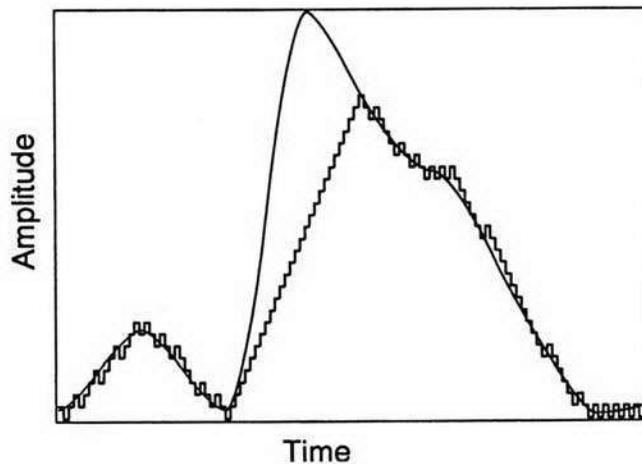


Figure 3.14. Delta Modulation. The jagged line is the delta modulated version of the smooth signal it encodes.

To compensate for these effects, **Continuously-Varying Slope Delta Modulation (CVSD)** uses a variable step size. Slope overload is detected as a series of consecutive “1” or “0” values. When this happens, the step size is increased to “catch up” with the input (see Figure 3.15). Note that the step size is increased at both the coder and the decoder, which remain synchronized because the decoder tracks the coder’s state by its output; the sequence of identical bit values triggers a step size change in both.

After the step size increases, it then decays over a short period of time. Both the encoder and decoder must use the same time constant for this decay; they remain in synchronization because the decoder uses the incoming bit stream as its clock. Quieting noise is reduced by detecting the quieting pattern of alternating “0” and “1” values and decreasing the step size to an even smaller value while this condition persists.

CVSD is an example of *adaptive* quantization, so called because the quantizer adapts to the characteristics of the input signal. During periods of low signal

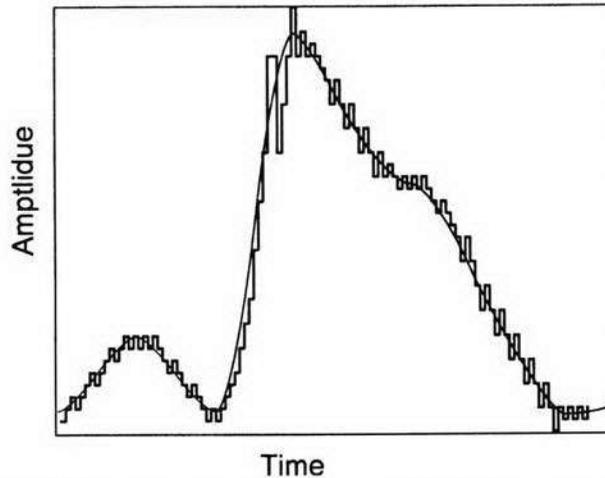


Figure 3.15. Quantizer step size increases for CVSD encoding to catch up with a rapidly increasing signal. The CVSD encoded version more closely tracks the original signal than does the delta modulation shown in Figure 3.14.

energy, a fine step size quantizer with a small range is employed, while a larger step size is substituted during higher amplitude signal excursions. In other words, at times when the signal is changing rapidly, there will be a larger difference between samples than at times when the signal is not changing rapidly. With CVSD these conditions are detected by monitoring the pattern of output bits from the quantizer.

A similar but more powerful encoder is **Adaptive Delta PCM (ADPCM)**, a popular encoding scheme in the 16,000 to 32,000 bits per second range. ADPCM is a Delta PCM encoder in that it encodes the difference between samples but does so using a method in which the same number of bits per sample can sometimes represent very small changes in the input signal and at other times represent much larger changes. The encoder accomplishes this by looking up the actual sample differences in a table and storing or transmitting just enough information to allow the decoder to know where to look up a difference value from the same table.

When it samples the current input value of the signal, an ADPCM coder uses two additional pieces of internal information in addition to the table to decide how to encode the signal. This information is a function of the history of the signal, and therefore is available to the decoder even though it is not transmitted or stored. The first stored value is the *state*, which is just a copy of the last value output from the coder; this is common to all difference encoders. The second value is the *predictor*, perhaps more appropriately called an *adapter*, which directly compensates for changes in the signal magnitude.

The key component of the ADPCM decoder is its output table. Although the difference signal is encoded in only four bits (including a sign bit to indicate an increasing or decreasing output value), the table is much “wider.” Each entry in

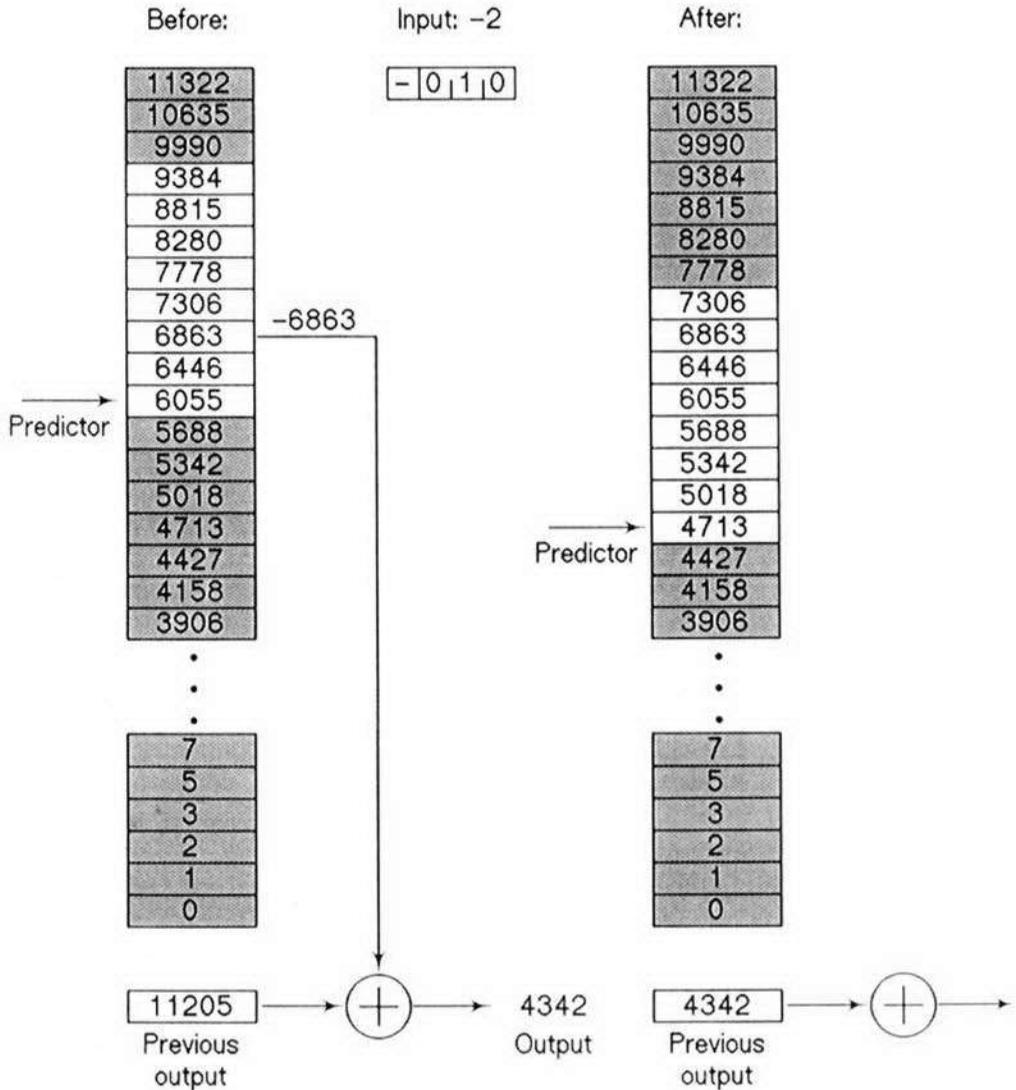


Figure 3.16. An ADPCM decoder.

the table may be eight or twelve bits giving a much larger dynamic range. At any moment, the three-bit difference value can be used to look up one of eight values from the table. These values are then added to or subtracted from the last output value giving the new output value. But the table has many more than eight entries. The predictor indicates where in the table the range of eight possible difference values should be chosen. As a function of the signal, the predictor can move to a coarser or finer part of the table (which is approximately exponential) as the signal amplitude changes. This movement within the lookup table is how adaptation occurs.

An ADPCM decoder is shown in Figure 3.16. For each input sample, the ADPCM decoder looks up a difference value that it adds to (or subtracts from) the previous output value to generate the new output value. It then adapts its own internal state. The difference value is found by table lookup relative to the predictor, which is just a base pointer into the table. This base pointer specifies the range of values which the encoder output can reference. Depending on the sign bit of the encoder output, the value found at the specified offset from the predictor is then added to or subtracted from the previous output value.

Based on another table, the input number is also used to look up an amount to move the base pointer. Moving the base pointer up selects a region of the lookup table with larger differences and coarser spacing; moving down has the opposite effect. This is shown on the right side of Figure 3.16 and is the adaptation mechanism for ADPCM.

Using an ADPCM encoding scheme with 4 bit samples at an 8 kHz sampling rate, it is possible to achieve speech quality virtually indistinguishable from that of 64,000 bit per second log PCM using only half of that data rate. Since many telephone channels are configured for 64,000 bits per second, ADPCM is used as part of a new approach to higher quality telephony that samples twice as often (16,000 times per second), thereby using the full digital telephone channel bandwidth to give a speech bandwidth of approximately 7.1 kHz. (The full algorithm is described in [Mermelstein 1975].) This has become an international standard because of its usefulness in telephone conference calls; the increased bandwidth is particularly helpful in conveying cues as to the identity of the speaker.⁹

In summary, waveform coders provide techniques to encode an input waveform such that the output waveform accurately matches the input. The simplest waveform coders are suitable for general audio encoding, although music sources place greater requirements on the encoder than speech. Due to the larger dynamic range of most music and its greater bandwidth, adequately digitizing music requires a larger sample size and a higher sampling rate. However, more sophisticated waveform coders take advantage of statistical properties of the speech signal to encode the signal in fewer bits without a loss of intelligibility. Because these methods depend on characteristics specific to speech, they are often not adequate for digitizing nonspeech audio signals.

Source Coders

In contrast to the waveform coders, that compress the speech signal by encoding the waveform, the source coders model the acoustical conditions that produced the signal. Source coders assume the sound was produced by the vocal tract and analyze it according to a source model. This model consists of an excitation component and a filter component corresponding to the vocal tract. Source decoders

⁹CCITT specification G.722. The **CCITT** is the International Telephone and Telegraph Consultative Committee, which defines international telecommunications standards.

then reconstruct the sound not by reproducing the waveform of the signal but rather by generating a different signal that has the same spectral characteristics as the original.

The most common source coder is **Linear Predictive Coding (LPC)**. LPC analysis requires somewhat expensive processing for the encoder but the decoder is fairly simple. An input signal is analyzed to determine whether it is periodic (whether the speech is voiced or not) and, if so, its pitch is computed. The coder then examines the spectrum of the input signal and determines a fixed number of resonances as parameters of a filter bank. The filter is an “all pole” model, which means it can reproduce only the peaks of the spectrum of the vocal tract. These parameters are predictor coefficients, which indicate how a sample at any moment in time depends on the previous samples. An overall gain value is also computed from the average magnitude or energy of the input. The coder updates this set of values at a regular rate, typically every 20 milliseconds.

The decoder (see Figure 3.17) models the vocal tract to produce a steady stream of digital waveform values. The model consists of a sound source and a filter and is controlled by the parameters derived by the encoder including pitch, voiced/unvoiced switch, gain, and filter coefficients. The source may be periodic with a range of possible pitches or it may be aperiodic. To output a waveform value, a new value is read from the input and scaled by a gain factor; then it undergoes a filtering process that can be represented by the weighted sum of the current sample and the previous N samples, where N is the number of predictor coefficients. Note that this system left alone would produce a continuous stream of steady output; the parameters determine what the output sounds like. But the model is updated with each new set of vocal tract parameters frequently enough to make it sound like speech.

The LPC decoder requires significantly less computation than the encoder, which must perform sophisticated analysis of the speech signal. This asymmetry may be useful in mass-market consumer applications, which store voice prompts

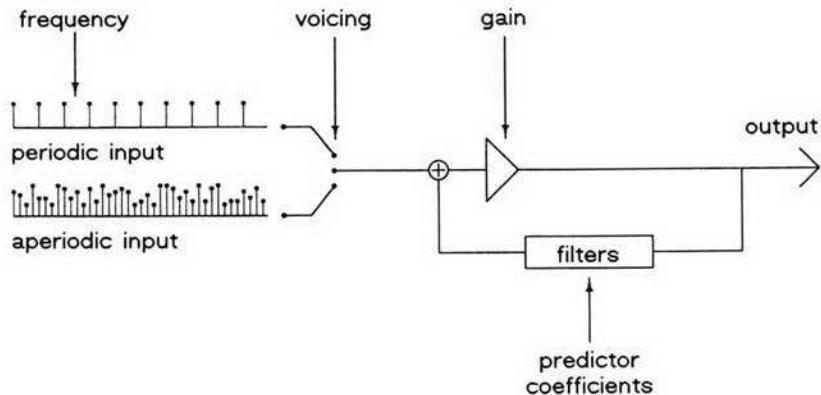


Figure 3.17. The LPC decoder models the vocal tract.

in read only memory (ROM). If the product need not encode speech but just play it, an expensive encoder is unnecessary. Such asymmetry is not an advantage for bidirectional applications such as telecommunications. Digitization algorithms that employ such different encoders and decoders are sometimes called **analysis-synthesis techniques**, which are often confused with text-to-speech synthesis; this is the topic of Chapter 5.

Speech can be encoded intelligibly using LPC at around 2400 bits per second. This speech is not of high enough quality to be useful in many applications and with the decreasing cost of storage has lost much of its original attractiveness as a coding medium. However, since LPC compactly represents the most salient acoustical characteristics of the vocal tract, it is sometimes used as an internal representation of speech for recognition purposes. In addition, a related source coding algorithm may emerge as the standard for the digital cellular telephone system.

A number of techniques have been developed to increase the quality of LPC encoding with higher data rates. In addition to the LPC signal, **Residually Excited LPC (RELP)** encodes a second data stream corresponding to the difference that would result between the original signal and the LPC synthesized version of the signal. This "residue" is used as the input to the vocal tract model by the decoder, resulting in improved signal reproduction.

Another technique is **Multipulse LPC**, which improves the vocal tract model for voiced sounds. LPC models this as a series of impulses, which are very short-duration spikes of energy. However, the glottal pulse is actually triangular in shape and the shape of the pulse determines its spectrum; Multipulse LPC approximates the glottal pulse as a number of impulses configured so as to produce a more faithful source spectrum.

The source coders do not attempt to reproduce the waveform of the input signal; instead, they produce another waveform that sounds like the original by virtue of having a similar spectrum. They employ a technique that models the original waveform in terms of how it might be produced. As a consequence, the source coders are not very suitable for encoding nonspeech sounds. They are also rather fragile, breaking down if the input includes background noise or two people talking at once.

CODER CONSIDERATIONS

The primary consideration when comparing the various speech coding schemes is the tradeoff between information rate and signal quality. When memory and/or communication bandwidth are available and quality is important, as in voice mail systems, higher data-rate coding algorithms are clearly desirable. In consumer products, especially toys, lower quality and lower cost coders are used as the decoder and speech storage must be inexpensive. This section discusses a number of considerations for matching a coding algorithm to a particular application environment.

Intelligibility

The primary consideration in choosing a coding scheme is that of intelligibility—*everyone* wants the highest quality speech possible. Some coding algorithms manage to reduce the data rate at little cost in intelligibility by using techniques to get the most out of each bit. In general, however, there is a strong tradeoff between speech quality and data rate. For any coding scheme there are likely to be options to use less bits for each value or to sample the audio less frequently at the cost of decreased intelligibility.

There are actually two concepts to consider: signal **error** and **intelligibility**. Error is a mathematically rigorously defined measure of the energy of the difference between two signals on a sample-by-sample basis. Error can be measured against waveforms for the waveform coders and against spectra for the source coders. But this error measure is less important than intelligibility for many applications. To measure intelligibility, human subjects must listen to reproduced speech and answer questions designed to determine how well the speech was understood. This is much more difficult than comparing two signals. The discussion of the intelligibility of synthetic speech in Chapter 9 describes some listening tests to measure intelligibility.

A still more evasive factor to measure is signal **quality**. The speech may be intelligible but is it pleasant to hear? Judgment on this is subjective but can be extremely important to the acceptance of a new product such as voice mail, where use is optional for the intended customers.

A related issue is the degree to which any speech coding method interferes with listeners' ability to identify the talker. For example, the limited bandwidth of the telephone sometimes makes it difficult for us to quickly recognize a caller even though the bandwidth is adequate for transmitting highly intelligible speech. For some voice applications, such as the use of a computer network to share voice and data during a multisite meeting, knowing who is speaking is crucial.

Editing

The ease with which a speech signal can be edited by cutting and splicing the sequence of stored data values may play a role in the choice of an encoding scheme. Both delta (difference) and adaptive coders track the current state during signal reproduction; this state is a function of a series of prior states, i.e., the history of the signal, and hence is computed each time the sound is played back. Simply splicing in data from another part of a file of speech stored with these coders does not work as the internal state of the decoder is incorrect.

Consider an ADPCM-encoded sound file as an example. In a low energy (quiet) portion of the sound, the predictor (the base pointer into the lookup table) is low so the difference between samples is small. In a louder portion of the signal the predictor points to a region of the table with large differences among samples. However, the position of the predictor is not encoded in the data stream. If samples from the loud portion of sound are spliced onto the quiet portion, the predictor will be incorrect and the output will be muffled or otherwise distorted

as the wrong differences will be accessed from the lookup table to produce the coder output.

With coders such as ADPCM predictable results can be achieved if cutting and splicing is forced to occur during silent parts of the signal; during silent portions the predictor returns to a known state, i.e., its minimum value. As long as enough low value samples from the silent period are retained in the data stream so as to force the predictor down to its minimum, another speech segment starting from the minimum value prediction (just after silence) can be inserted. This precludes fine editing at the syllable level but may suffice for applications involving editing phrases or sentences.

Adaptation is not an issue with PCM encoding as the value of each sample is independent of any prior state. However, when pieces of PCM data are spliced together, the continuity of the waveform is momentarily disrupted, producing a “click” or “pop” during playback. This is due to changing the *shape* of the waveform, not just its value, from sample to sample. The discontinuity is less severe when the splice takes place during low energy portions of the signal, since the two segments are less likely to represent drastically different waveform shapes. The noise can be further reduced by both interpolating across the splice and reducing the amplitude of the samples around the splice for a short duration.

A packetization approach may make it possible to implement cut-and-paste editing using coders that have internal state. For example, LPC speech updates the entire set of decoder parameters every 20 milliseconds. LPC is really a stream of data packets, each defining an unrelated portion of sound; this allows edits at a granularity of 20 milliseconds. The decoder is designed to interpolate among the disparate sets of parameter values it receives with each packet allowing easy substitution of other packets while editing.

A similar packetization scheme can be implemented with many of the coding schemes by periodically including a copy of the coder’s internal state information in the data stream. With ADPCM for example, the predictor and output signal value would be included perhaps every 100 milliseconds. Each packet of data would include a header containing the decoder state, and the state would be reset from this data each time a packet was read. This would allow editing to be performed on chunks of sound with lengths equal to multiples of the packet size.

In the absence of such a packetization scheme, it is necessary to first convert the entire sound file to PCM for editing. Editing is performed on the PCM data and then is converted back to the desired format by recoding the signal based on the new data stream.

Silence Removal

In spontaneous speech there are substantial periods of silence.¹⁰ The speaker often pauses briefly to gather his or her thoughts either in midsentence or

¹⁰Spontaneous speech is normal unrehearsed speech. Reading aloud or reciting a passage from memory are not spontaneous.

between sentences. Although the term "sentence" may be used fairly loosely with spontaneous speech, there are speech units marked by pauses for breath that usually correspond roughly to clauses or sentences. Removing these silences during recording can save storage space. Such techniques are commonly applied to almost any encoding technique. For example, silence removal is used with most voice mail systems regardless of the encoding scheme. The amount of storage space saved by silence removal is directly proportional to the amount of speech that was pauses.

A form of silence removal is actually used in many telephone networks. There is no need to keep a transmission channel occupied transferring the audio data associated with the frequent pauses in a conversation. By detecting these pauses, the channel may be freed to carry another conversation for the duration of the pause. When one of the parties starts speaking again, the channel must be restored to transmit the conversation. In practice, a small number of channels are used to carry a larger number of conversations; whenever a new spurt of speech occurs during one of the conversations, a free channel is assigned to it.

This channel-sharing scheme was originally implemented as an analog system known as **Time Assigned Speech Interpolation (TASI)** [O'Neill 1959] to carry an increased number of calls over limited-capacity, trans-Atlantic telephone cables. Today, digital techniques are employed to implement "virtual circuits," so called because the actual channel being used to carry a specific conversation will vary over time. The channel refers to a particular slot number in a digital data stream carrying a large number of conversations simultaneously. During silence, the slot associated with the conversation is freed and made available to the next conversation that begins. When the first conversation resumes, the slot it formerly occupied may be in use, in which case it is assigned to a new slot. These adjustments are transparent to equipment outside of the network. The number of channels required to carry the larger number of conversations is a property of the statistical nature of the speaking and silence intervals in ordinary conversations and is an important aspect of network traffic management.

During a telephone conversation, no time compression is achieved by silence removal; although audio data may not be transmitted across the network during silences, the listener must still wait for the talker to resume. In a recorded message with pauses removed, less time is required for playback than transpired during recording. Silence removal is generally effective, but some of the pauses in speech contain meaning; part of the richness of the message may be the indication of the talker's thought process and degree of deliberation implied by the location and duration of the pauses.

Several techniques can be used to reconstruct the missing pauses. Instead of simply removing the pauses from the stored data, a special data value may be inserted into the file indicating the presence of a pause and its duration. During playback this information indicates to the decoder to insert the right amount of silence. Since the ambient (background) noise of the recording is replaced by absolute silence, an audible discontinuity is noticeable. Another technique involves recording a small amount of the background noise and then playing it

over and over to fill the reconstructed pauses. It is necessary to capture enough of the silence that the listener cannot easily hear the repetitive pattern during playback.

Another difficulty with silence removal is proper detection of the beginning, or onset, of speech after silence. The beginnings and endings of speech segments are often of low amplitude, which may make them difficult to detect, yet they contain significant phonetic information. Endings can be easily preserved simply by recording a little extra “silence” at the end of each burst of speech. Beginnings are harder; the encoder must buffer a small amount of the signal (several hundred milliseconds) so that when speech is finally detected this recently-saved portion can be stored with the rest of the signal. This buffering is difficult enough that it is often not done, degrading the speech signal.

Time Scaling

Because speech is slow from the perspective of the listener, it is desirable to be able to play it back in less time than required to speak it originally. In some cases one may wish to slow down playback, perhaps while jotting down a phone number in a rapidly-spoken message. Changing the playback rate is referred to as **time compression** or **time-scale modification**, and a variety of techniques of varying complexity are available to perform it.¹¹

Skipping pauses in the recording provides a small decrease in playback time. The previous section discussed pause removal from the perspective of minimizing storage or transmission channel requirements; cutting pauses saves time during playback as well. It is beneficial to be selective about how pauses are removed. Short (less than about 400 millisecond) **hesitation pauses** are usually not under the control of the talker and can be eliminated. But longer **juncture pauses** (about 500 to 1000 milliseconds) convey syntactic and semantic information as to the linguistic content of speech. These pauses, which often occur between sentences or clauses, are important to comprehension and eliminating them reduces intelligibility. One practical scheme is to eliminate hesitation pauses during playback and shorten all juncture pauses to a minimum amount of silence of perhaps 400 milliseconds.

The upper bound on the benefit to be gained by dropping pauses is the total duration of pauses in a recording, and for most speech this is significantly less than 20% of the total. Playback of nonsilent portions can be sped up by increasing the clock rate at which samples are played, but this produces unacceptable distortion. Increasing the clock shifts the frequency of speech upwards and it becomes whiny, the “cartoon chipmunk” effect.

A better technique, which can be surprisingly effective if implemented properly, is to selectively discard portions of the speech data during playback. For example,

¹¹For a survey of time-scale modification techniques, see [Arons 1992a].

for an increase in playback rate of 20%, one could skip the first 12 milliseconds of every 60 milliseconds of data. This was first discussed by Maxemchuk [Maxemchuk 1980] who refined it somewhat by attempting to drop lower-energy portions of the speech signal in preference to higher energy sections. This technique is rather sensitive to the size of the interval over which a portion of the sound is dropped (60 milliseconds in the above example); this interval should ideally be larger than a pitch period but shorter than a phoneme.

Time compression by this method exhibits a burbling sound but can be surprisingly intelligible. The distortion can be somewhat reduced by smoothing the junctions of processed segments, by dropping and then increasing amplitude rapidly across the segment boundaries, or by performing a cross-fade in which the first segment decays in amplitude as the second segment starts up.

A preferred method of time compression is to identify pitch periods in the speech signal and remove whole periods. To the extent that the signal is periodic, removing one or more pitch periods should result in an almost unchanged but time-compressed output. The most popular of these time domain algorithms is **Synchronous Overlap and Add (SOLA)** [Roucos and Wilgus 1985]. SOLA removes pitch periods by operating on small windows of the signal, shifting the signal with respect to itself and finding the offset with the highest correlation, and then averaging the shifted segments. Essentially, this is pitch estimation by autocorrelation (finding the closest match time shift) and then removal of one or more pitch periods with interpolation. SOLA is well within the capacity of many current desktop workstations without any additional signal processing hardware.

Time compression techniques such as SOLA reduce the intelligibility of speech, but even naive listeners can understand speech played back twice as fast as the original; this ability increases with listener exposure of 8 to 10 hours [Orr *et al.* 1965]. A less formal study found listeners to be uncomfortable listening to normal speech after one-half hour of exposure to compressed speech [Beasley and Maki 1976].

Robustness

A final consideration for some applications is the robustness of the speech encoding scheme. Depending on the storage and transmission techniques used, some data errors may occur, resulting in missing or corrupted data. The more highly compressed the signal, the more information is represented by each bit and the more destructive any data loss may be.

Without packetization of the audio data and saving state variables in packet headers, the temporary interruption of the data stream for most of the adaptive encoders can be destructive for the same reasons mentioned in the discussion of sound editing. Since the decoder determines the state of the encoder simply from the values of the data stream, the encoder and decoder will fall out of synchronization during temporary link loss. If the state information is transmitted periodically, then the decoder can recover relatively quickly.

Another aspect of robustness is the ability of the encoder to cope with noise on the input audio signal. Since the encoders rely on properties of the speech signal

to reduce their bit rates, inputs that fail to obey these constraints will not be properly encoded. One should not expect to encode music with most of these encoders, for example. LPC is an extreme case as it is a source coder and the decoder incorporates a model of the vocal tract.

SUMMARY

This chapter has discussed digitization and encoding of speech signals. Digitization is the process whereby naturally analog speech is sampled and quantized into a digital form amenable to manipulation by a computer. This may be likened to the previous chapter's description of how the ear transduces variations in air pressure on the ear drum to electrical pulses on neurons connected to the brain. Digitization stores a sound for future rote playback, but higher level processing is required to understand speech by recognizing its constituent words.

This chapter has described several current speech coding schemes in some detail to illustrate how knowledge of the properties of speech can be used to compress a signal. This is not meant to be an exhaustive list, and new coding algorithms have been gaining acceptance even as this book was being written. The coders described here do illustrate some of the more common coders that the reader is likely to encounter. The design considerations across coders reveal some of the underlying tradeoffs in speech coding. Basic issues of sampling rate, quantization error, and how the coders attempt to cope with these to achieve intelligibility should now be somewhat clear.

This chapter has also mentioned some of the considerations in addition to data rate that should influence the choice of coder. Issues such as coder complexity, speech intelligibility, ease of editing, and coder robustness may come into play with varying importance as a function of a particular application. Many of the encoding schemes mentioned take advantage of redundancies in the speech signal to reduce data rates. As a result, they may not be at all suitable for encoding nonspeech sounds, either music or noise.

Digital encoding of speech may be useful for several classes of applications; these will be discussed in the next chapter. Finally, regardless of the encoding scheme, there are a number of human factors issues to be faced in the use of speech output. This discussion is deferred to Chapter 6 after the discussion of text-to-speech synthesis, as both recorded and synthetic speech have similar advantages and liabilities.

FURTHER READING

Rabiner and Schafer is the definitive text on digital coding of speech. Flanagan *et al.* [1979] is a concise survey article covering many of the same issues in summary form. Both these sources assume familiarity with basic digital signal processing concepts and mathematic notation.