Internet Telephony: Architecture and Protocols

Jonathan Rosenberg
Bell Laboratories
jdrosen@bell-labs.com

Henning Schulzrinne
Columbia University
hgs@cs.columbia.edu

September 9, 1999

Chapter 1

Multimedia

As we have mentioned, despite its name, Internet telephony is about the exchange of multimedia - speech, video, and even data, between participants in real time over the Internet. In order to be transported over the Internet, the multimedia must be represented in a digital format. Of course, speech and video are not naturally encountered in digital format. They must be acquired through a microphone or camera, usually resulting in an analog representation, and then from there converted to digital.

This digitization process is often referred to as *coding*, *encoding*, or *compression*, although these really represent different concepts. Coding and encoding refer to the conversion from analog to digital; compression refers to a digital to digital conversion which results in a reduction in the in the number of bits required to represent the multimedia. Since the coding and compression steps are often combined into one, the terms are frequently used interchangeably. A *codec* is a method of compressing the media, and consists of an encoder and decoder. Codecs are frequently standardized (in particular, the decoder is usually standardized, allowing people to develop their own encoders), to allow interoperability.

There are many different ways in which both speech and video can be encoded for transport on the Internet. The various schemes differ along a number of axes [1]:

Bitrate Bitrate refers to the number of bits per second required to represent the media stream. The larger the bitrate, the more the amount of information sent, and in general, the better the quality of the recovered analog signal at the receiver. However, increased bitrates result in more usage of network resources.

Quality It is difficult to objectively define the quality of a speech or video signal. However, the act of compression and reconstruction results in a reduction of the overall quality of the media. This can be observed through the introduction of *artifacts*, or specific types of errors, into the signal. Speech artifacts include tinny sound, audible clicks, or a muffled characteristic. Video artifacts include jerky motion, blurring, "buzzing" around edges, poor color, and blockiness. Different codecs yield different quality.

Delay The process of encoding and decoding incurs a delay. This delay, also known as the *algorithmic delay*, is not due to limited processing resources, but to fundamental delays introduced by the codec itself.

Robustness to Loss This is characteristic that is particularly important for Internet telephony. The Internet will frequently drop packets from users, resulting in missing pieces of data at the receiver. The media must be reconstructed in the absence of this data. The reconstruction process usually results in an errored signal due to this loss. Different codecs are more or less susceptible to packet losses. In general, the lower the bitrate of the codec, the more susceptible it is to loss. This is a natural consequence of information theory. The more compression that is applied, the more useful information is conveyed in each packet. Therefore, the loss of a single packet causes a loss of more information, and a reduction in reconstructed speech quality.

Complexity The process of encoding and decoding requires computational resources. The amount that is required depends on the codec, and in some cases on the content as well.

Tandem Performance Often, in complex systems, the speech or video may be decoded and re-encoded numerous times. This process is known as *tandeming*. Certain codecs perform worse than others in this scenario.

Non-Speech Performance . Speech codecs sometimes need to encode data that is not speech. This is especially true in interactive environments, like Internet telephony, where the data being encoded is being presented by the user in real time. Some of the signals which sometimes are encoded include:

Music When a user calls a help desk, they are often put on-hold, and listen to "muzak". The music is encoded by the speech

codec. Music contains much different frequency components, and its compression with a speech codec usually yields poor quality.

Voiceband Data Modem signals are frequently sent over telephone lines. For phone to phone Internet telephony (see Section XXX), these modem signals will be sent over the Internet using a speech codec.

Fax Like modem signals, fax signals may also be sent using a speech codec in an Internet telephony phone to phone environment.

Tones When a user presses a number on their phone, a tone is generated. These tones contain two pure frequencies, and for this reason the tones are called *Dual Tone Multi-Frequency*, or DTMF for short. These tones may be encoded with the speech codec.

Some codecs do a good job in encoding non-speech signals, and others do a poor job. The problem can be avoided in systems which determine that the signal is not speech, and send it by some other means. See section XXX for a discussion of how tones are handled.

Choosing the "right" audio or video codec for an Internet telephony application depends on the desired characteristics. For high quality conferencing applications, high quality and low susceptibility to loss may be more important than bitrate. However, for consumer Internet telephony, low bitrate may be more important. Fortunately, its not neccesary to pick *one* codec for all applications, or even for a single application. On the Internet, the multimedia is transmitted directly between the senders and receivers through IP routers which are *application unaware*. The routers are only concerned with moving data packets from one IP address to another. Whether the packets contain speech, video, or email, is of no concern to the router. Because of this, its only necessary that the parties involved in the Internet telephony call use the same codec for that call. For a different call, a different codec can be used. Part of the call setup process, called *capabilities exchange*, allows the participants to determine a set of codecs acceptable to all. We discuss this further in section XXX.

Even though the codec can change from call to call, the codec must be one understood by all participants. To maximize interoperability, industry groups have tried to standardize *baseline* codecs, which are codecs that every Internet telephony application should be able to understand. Applications can understand others too; but ensuring that a baseline codec exists means everyone can always talk to anyone else. Unfortunately, Intellectual Property and patent issues have made such a selection a difficult process.

Because of the flexibility Internet telephony provides in allowing different codecs to be used, and because of the different properties of these codecs, we review some of the basic concepts in compression here. A full treatment is beyond the scope of this book; entire texts have been dedicated to speech and video separately. The reader is referred to XXX (Peter Kroon, Arun's book, Haskells book, etc.) for more detailed information.

1.1 Speech

In this section, we focus on speech. We first discuss the properties of speech - how its generated and what its components are. We then discuss the process of digitizing speech - converting it from an analog waveform to a basic digital representation, using Pulse Code Modulation (PCM). We then discuss how the PCM encoded speech is further compressed for transport on the Internet.

1.1.1 Properties of Speech

A snapshot of a typical speech waveform is shown in Figure INSERT1, which represents a female speaking the word "cheat". There are clearly two components to this signal. The initial, "non-periodic" component represents the "ch" sound, and the component which follows represeants the "ea" sound. The former component is known as an *unvoiced* sound, since the vocal cords are not used to generate it, while the latter is *voiced*, since they are used.

[INSERT1: Copy Figure 1, chapter 1 of Kleijn and Paliwal]

Speech generally has frequency components in the 300 to 3400 Hz range. The power spectrum (which represents the contribution of various frequencies to a signal) for the waveform in Figure INSERT1 is shown in Figure INSERT2. The left graph is the power spectrum for the unvoiced component, and the one on the right is for the voiced component. Notice that the graph on the right exhibits strong frequency components at fixed intervals apart, modulated by a spectral envelope. It is this characteristic of speech which is taken advantage of in many speech coding systems.

[INSERT2 Copy Figure 2, chapter 1 of Kleijn and Paliwal]

During a typical human conversation, one party generally talks at a time, while the other remains silent. The speech from a single participant therefore exhibits periods of talking (called *talkspurts*), followed by periods of inactivity, called *silence periods*. In fact, there are even finer grained talkspurt and silence periods in human speech. Brief pauses exist between sentences, even briefer ones between words, and extremely brief ones between

syllables. When the speech is transported on the Internet, these silence periods can be taken advantage of. Little or no information can be sent during these periods. This reduces network resources.

Extensive studies were conducted during the late sixties and early seventies to explore the properties of these silence periods and talkspurts. It was shown that the duration of these periods could be represented by exponentially distributed random variables. Silence periods were 1 to 1.6 seconds on average, and talkspurts 800ms to 1.2 seconds, on average. This allowed speech to be represented by a two-state Markov process, as shown in Figure 1.1. Speech moves between the talking and silent periods. By sending nothing during silence, the number of packets transmitted can be reduced by nearly 40%.

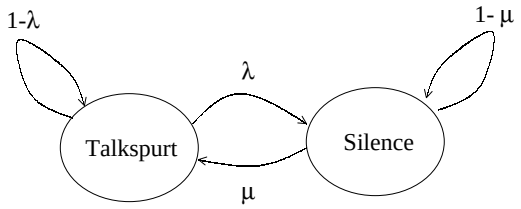

Figure 1.1: Markov Model of Human Speech

1.1.2 Digitization of Speech

The digitization of speech is a multi-step process. The speech is first low pass filtered, removing the high frequency components that would otherwise show up as aliasing noise. After that, the speech is sampled. The sampling process, shown in Figure 1.2, picks values of the speech waveform at intervals Δ seconds apart. According to the Nyquist sampling theorem, $1/\Delta \geq 2f_{max}$,

where f_max is the highest frequency component in the signal. Since speech contains frequency components up to around 3.4 khz, the speech is usually sampled every 125 microsends, yielding a sampling rate of 8 kHz.

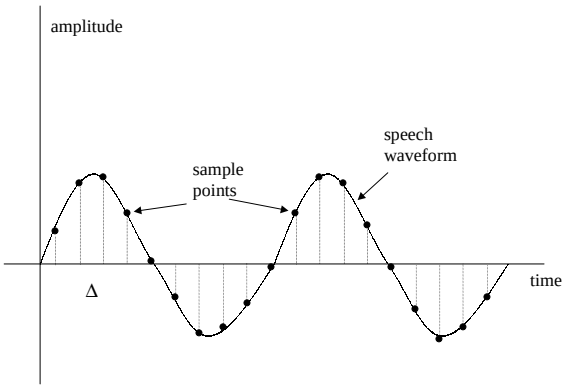

Figure 1.2: Sampling Process

The sampled analog values must then be *quantized*. A certain number of bits are used to represent each analog value. With N bits, only 2^N possible analog values can be represented. As such, each analog value that is sampled must be mapped to one of the 2^N quantized values. The function used for this mapping is called a *quantizer*, an example of which is shown in Figure 1.3. Notice the stair-like nature of this function. A large range of input values all map to a single output. This is the exact property that is required. Since, in this example, the function has only eight possible outputs, they can be represented using three bits.

The process of performing this mapping comes at a cost. At the decoder, each of the 2^N quantized values are converted to an analog value. This analog value will not be the same as the original analog value. The difference depends on how far away the original value is from the reconstructed value at the decoder. The reconstructed value is usually the value in the middle of the "tread" for each step in the quantizer function. This means the difference (also known as *quantization error*, is at most one half of the width of each tread. As the number of bits N increases, the number of treads in

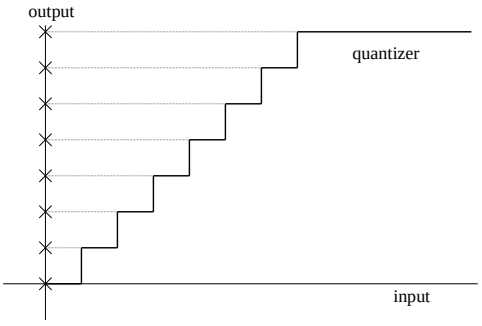

Figure 1.3: Linear Quantizer

the quantizer increases, and the width of each decreases, resulting in better reconstruction, but more bits.

For accurate representation of speech, a *linear quantizer* with $N = 16$ is generally used. A linear quantizer is one in which the width of each step in the quantizer function is the same, so that the amount of quantization error in each sample has a range which does not depend on the value of the sample. By using $N = 16$, the digital speech requires a bitrate of 16 bits per sample, times 8000 samples per second, yielding 128 kb/s. The process of sampling, followed by quantization of the sampled values, is often referred to as *Pulse Code Modulation*, or PCM.

It turns out that a more intelligent choice of the quantizer can reduce this rate. Humans are more sensitive to quantization errors when the amplitude of the signal is lower. When the amplitude is higher, the quantization noise is less noticeable. To take advantage of this, the spacing between quantized values can be made non-linear. The spacing is made large at larger values, and small at smaller values. In particular, the use of a logarithmic scale has been found to be effective. This allows only 8 bits to be used instead of 16, with nearly equal quality.

Two logarithmic quantizers are in general use today. These are known as *μ-law* and *A-law*. The μ-law quantizer is used in North America and Japan,

while A-law is used everywhere else. Both quantizers are standardized in the International Telecommunications Union (ITU) G.711 recommendation [2].

1.1.3 Compression of Speech

The process of speech compression takes an uncompressed, digital representation of the speech, and produces another digital representation that requires fewer bits. Broadly, there are two types of compression - *lossless* and *lossy*. Lossless compression guarantees that no information is lost. If data (here, the speech signal) is compressed and decompressed, the resulting data is identical, bit by bit, when compared to the original. Lossy compression causes data to be lost, but the reconstructed signal still sounds close to the original. Not surprisingly, lossy compression yields much lower bitrates, and most speech codecs in use are lossy.

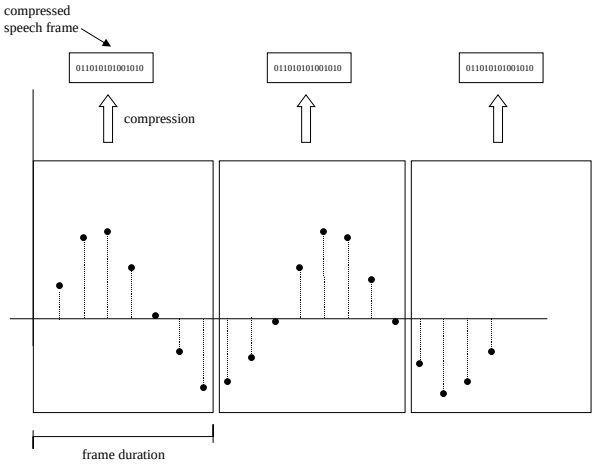

Figure 1.4: Breaking Speech into Frames

To perform the compression, most speech codecs start by taking the speech, and breaking it into fixed-length sections, called *frames*. As shown in Figure 1.4, each frame is then compressed as a unit, resulting in a set of bits which represent the frame. These compressed frames are then transmitted to the decoder. Some speech coders use *subframes*, which are even smaller

partitions of each frame. When the frame is compressed, it contains data common to both subframes, along with data particular to the reconstruction of each subframe.

Some codecs also use a method called *lookahead*. The speech is still broken into frames, but the compression process for a frame requires information from a speech samples in the future. This is illustrated in Figure 1.5. The compressed frame represents the speech between the first two arrows, but data between the second and third is used to generate some of the parameters in the frame.

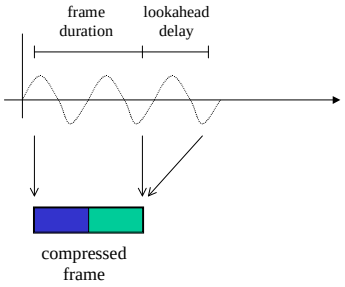

Figure 1.5: Lookahead in speech compression

Frame sizes and lookahead durations are important since they determine the algorithmic delay of the codec. If a codec has a frame size of 20 ms, the encoder must collect 20 ms worth of speech before any data can be sent to the decoder. This introduces a delay of 20 ms into the system. A codec that has a frame size of 30 ms, with a 10 ms lookahead delay, will result in an algorithmic delay of 40 ms.

After breaking the speech into frames, many of the modern speech codecs apply a principle called *Linear Predictive Analysis by Synthesis*, or LPAS, to achieve compression. There are two components in this principle. The first is *linear prediction*. The idea is that the speech signal can be represented by the output of a linear filter with some specific input, known as the *excitation*, as

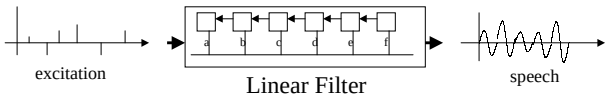

Figure 1.6: Linear Filtering for Speech Synthesis

shown in Figure 1.6. Instead of sending the speech itself, the encoder sends the filter structure and a representation of the excitation to the decoder. The encoder is therefore tasked with computing the filter and determining the excitation. For a variety of reasons, an all-pole linear filter is generally used. Mathematical methods (specifically, computation of the autocorrelation of the signal, followed by solution of a matrix equation known as the *Yule-Walker* Equation) [**?**] are often used to determine the coefficients of this filter. However, the excitation is determined through *analysis by synthesis*, depicted in Figure 1.7. In this process, the input is determined by searching through the possible inputs, and for each one, synthesizing the speech output by passing the input through the filter, and then computing some measure of the error. This error measure is usual *perceptual*, taking into account human perception of speech. The input which produces the smallest error is then chosen, and transmitted to the decoder. Since the speech signal is analyzed (i.e., it is broken down into an excitation and a filter) by synthesizing the speech signal over a variety of inputs, the overall procedure is known as analysis by synthesis.

The differences between most LPAS speech codes is the way the excitation is represented. One class of codecs, known as *Codebook-Excited Linear Predictive*, or CELP, use a codebook of possible excitations. Both the en-

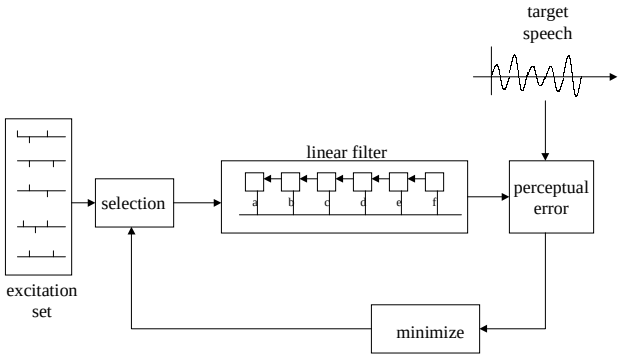

Figure 1.7: Linear Predictive Analysis by Synthesis

coder and decoder have a copy of this codebook. Each entry contains a possible excitation. Rather than sending the excitation, the index of the codebook entry is sent instead.

To improve performance, both *postfilters* and *prefilters* are often used. Postfilters are operations applied to the speech after it has been decompressed. These are usually filters that clean-up the speech, removing some of the effects of the encoding process. A prefilter, on the other hand, is used by the encoder. It prepares the signal for compression, modifying it in a way that makes the compression and decompression process result in a signal that sounds better.

With an understanding of these basic concepts, we can discuss some of the codecs in common usage for Internet Telephony.

G.726 - ADPCM

Adaptive Differential Pulse Code Modulation (ADPCM) is specified in ITU Recommendation G.726 [3]. It represents the merging of G.721 and G.723, prior standards for ADPCM. It was standardized in December of 1990. G.726 actually allows for the encoding of speech at a number of bitrates, including 40, 32, 24, and 16 kb/s.

ADPCM is not a CELP coder, nor does it break the speech into blocks. For each sample (every 125 us), it computes an estimate of the sample based on previous transmitted data (so that the decoder can also compute this estimate). The estimate is, in fact, a linear filter, containing both poles and zeroes. The difference between the estimate and the actual value is then quantized, and this quantized value is sent to the decoder. ADPCM uses a steepest descent algorithm to adapt the estimation filter. The quantizer used on the difference value is also adapted. ADPCM achieves differing bit rates by using a different number of bits for the quantization. When running at 40 kb/s, 5 bits are used, for 32 kb/s, 4 bits, for 24, 3 bits, and for 16 kb/s, 2 bits.

ADPCM is advantageous in that there is no algorithmic delays at all, which is ideal for interactive applications like Internet telephony. It has also been engineered to support voiceband data (i.e., modem signals) and fax. At the 40 kb/s coding rate, modems up to 12 kb/s suffer no degradation in performance. The maximum rate decreases as the ADPCM rate drops. ADPCM also transmits tones and fax signals with little degradation.

The disadvantage to ADPCM is its high bitrate. Good speech quality is achieved only at 32 kb/s and above. There is a noticeable effect at the 24 and 16 kb/s rates. Much better codecs exist that give good performance even down to 6 kb/s. ADPCM is also sensitive to bursts of loss, which makes its usage in the Internet dubious at best.

G.728 - LD-CELP

The Low-Delay Codebook Excited Linear Predictor (LD-CELP) codec was standardized by the ITU in 1992, and is codified in recommendation G.728 [4]. As its name implies, LD-CELP is a low delay speech codec. It breaks the speech into frames of 5 samples, yielding an algorithm delay of 625 μs. Each frame is sent using 10 bits, resulting in a rate of 16 kb/s.

LD-CELP achieves its low delays by using a technique known as *backwards prediction*. The other CELP codecs described in this chapter send the coefficients for the synthesis filter directly to the decoder. This requires the encoder to build up a sufficient frame size to accurately compute the coefficients. LD-CELP, on the other hand, does not transmit the coefficients. The coefficients are computed based on the decoded output speech. Since the decoded output speech is available at both encoder and decoder, no information needs to be transmitted. ADPCM works in a similar fashion, adapting the filter coefficients based on past speech output.

LD-CELP assumes a codebook with 1024 entries as the excitation. The

encoder searches the entries to find the one which yields the best match to the input speech signal. The only information sent to the decoder is the index of the codebook vector.

[LD-CELP for IP tel?]

G.729 - CS-ACELP

Conjugate Structure Algebraic Code Excited Linear Prediction (CS-ACELP) was standardized by the ITU in 1996, and codified in recommendation G.729 [5]. It generates speech at a rate of 8 kb/s. The uncompressed speech is broken into frames 10 ms in duration, and a lookahead of 5 ms is used, for an algorithmic delay of 15 ms. Each frame is additionally broken into two 5 ms subframes. Each frame is sent using 80 bits.

The LD-CELP algorithm first applies a prefilter to eliminate low-frequency components. The encoder then computes the filter coefficients, using speech data 5ms into the future (the lookahead delay), speech from the current frame, and speech from 15 ms in the past. These coefficients are then sent to the decoder.

The excitation for the filter is composed of two parts. The first is an *adaptive codebook* contribution. This contribution is actually a vector of excitation extracted from past excitations. By using excitation from the past, the encoder is able to model the natural harmonic components in the speech. The delay, or amount of distance in the past from which the excitation is chosen, is known as the *pitch*. The pitch, along with a gain factor used to multiply the past excitation before using it, are sent to the decoder. The second contribution to the excitation is a *fixed codebook* contribution. This consists of a series of pulses and gain factors stored in a codebook. The encoder searches among the codebook entries until it finds the best one, and this is transmitted to the decoder as well.

G.729 additionally describes procedures for handling *frame erasures*, which occur when an entire frame is lost. On the Internet, this happens when a packet containing the frame is lost. These procedures interpolate the missing information, improving the quality of the speech.

A number of annexes have been developed for G.729. Annex A [6] defines a low-complexity version of G.729. Annex B [7] defines a silence suppression scheme. This scheme employs a *voice activity detector*, or VAD, which determines, for each frame, whether there is speech or silence. Studies have shown that it is disturbing to users for the output of the speakers to stop during silence periods. Rather, *comfort noise*, which represents the background noise, is preferred. In Annex B, the encoder computes parameters

that describe this comfort noise. They are sent to the decoder when they are sufficiently different from those in the previous frame. The decoder uses these parameters to generate the comfort noise. If no parameters are sent for the current frame, the parameters from the previous are used.

Annex D [8] specifies a modification of the original G.729 algorithm which runs at 6.4 kb/s, and annex E [9] specifies a different modification which permits 11.8 kb/s. The increase in rate results in better coding of music and non-speech signals.

G.729 is well suited for Internet telephony. Its low rate is good for modem dialup users where bandwidth is limited. It is quite resilient to packet loss (which translates into frame erasures), and the speech quality is very good.

G.723.1

G.723.1 (which has no formal acronym) describes a dual-rate speech codec. The codec can generate compressed speech at a rate of either 5.3 or 6.3 kb/s. It breaks the speech into frames 30 ms in duration, and uses a 7.5 ms lookahead, resulting in an algorithmic delay of 37.5 ms. G.723 allows the encoder to switch between 5.3 and 6.3 kb/s on a frame by frame basis. The higher rate has a better quality, of course. This allows engineers a tradeoff between rate and quality.

1.2 Video

In the broader definition of the term, Internet telephony can include all sorts of multimedia, including video. The range of bitrates for video codecs is very wide. It ranges from as low as 8 kb/s for "talking-head" conferencing codecs, to nearly 20 Mb/s for high definition television. The high rate, television quality codecs generally do not run in real time, and are used mainly for broadcast applications. Although broadcast systems form a component of a complete Internet telephony system, we focus instead on the lower rate, interactive codecs. The core features are similar across all of the codecs, however.

1.2.1 Properties of Video

Digital video consists of a number of *frames*, each frame representing a complete picture. The frames are shown in sequence at a particular *frame rate* anywhere between 5 and 30 frames per second, yielding the perception of motion video to a human observer. Five frames per second is very jerky,

and can be disturbing to watch. Thirty frames per second is considered very good for smooth motion video. Television signals in the United States have frame rates of 30 frames per second. In fact, televisions break each frame into two fields, an odd and even field. Odd fields contain the odd lines of the picture, and even fields the even lines. Instead of presenting the frame all at once, one field is shown, and then 1/60 second later, the other field. The result is a rate of 60 fields per second.

Each frame is composed of *pixels*, or picture elements, which are tiny dots that, when put together, compose the frame. Each pixel is a specific color. Representation of color in a digital format can be done in a number of ways. In general, it is composed of three components - the amount of red, the amount of green, and the amount of blue (RGB). A pixel has a certain amount of each color, ranging from zero to a maximum amount. The amount of each can therefore be represented with a binary value. Eight bits is generally sufficient, for a total of 24 bits overall. 24 bit color is often referred to as *true-color*.

The amount of red, green, and blue can be considered as a vector in a three dimensional space. Like any vector system, different basis vectors can be used to represent the color point. A very common format, known as YUV, represents each pixel by a luminance (the Y component), and two color difference components (U and V), also known as chrominance. The Y component represents the brightness of the pixel, and the chrominance its color. Setting the chrominance to zero converts a pixel to black and white. YUV is generally the preferred representation for pixels. It turns out that most of the high frequency energy in a picture is in the luminance component. The chrominance is very low-frequency. It can be transmitted with fewer bits than the luminance with no noticeable effect.

1.2.2 Compression of Video - MC-DCT

Many techniques have been proposed for the compression of video. These include a variety of frequency decompositions (known as sub-band coding), and fractal based compression, among others (need references here XXX). However, nearly all commercial video compression systems are based on the same underlying principle - the Motion Compensated Discrete Cosine Transform (MC-DCT).

Discrete Cosine Transform

The process starts by breaking the image into a number of *macroblocks*. Each macroblock is a 16x16 array of pixels from the original image. The chrominance components (U and V) are subsampled, so that an 8x8 *chrominance block* is used to represent each color component for the macroblock. The luminance is broken into four 8x8 *blocks*. This enables a macroblock to be represented by a total of six 8x8 blocks, four luminance and two chrominance, as shown in Figure 1.8.

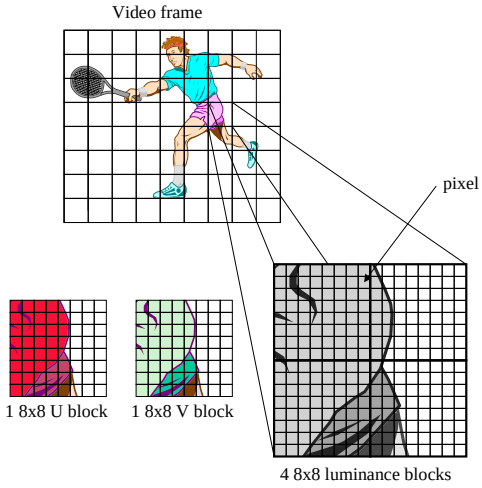

Figure 1.8: Breaking an image into macroblocks

Instead of sending each block directly, the block is transformed into a different representation, by applying the Discrete Cosine Transform (DCT). The DCT converts in image into an equivalent representation. In this representation, the image is composed of a sum of *basis images*. These basis images are cosines in the horizontal and vertical directions. There are 64 basis images, each differing in the frequency of the cosines. The basis images are shown in Figure INSERT3.

[INSERT3 Figure 3.3 of Riley,Richardson]

Each basis image can be characterized by its horizontal and vertical frequency. There are eight different frequencies possible in each direction. If we number the horizontal frequencies u from 0 to 7, and the vertical

frequencies v from 0 to 7, each basis image $B_{u,v}(i,j)$ can be defined as:

$$B_{u,v}(i,j) = cos\left(\frac{(2i+1)u\pi}{16}\right) cos\left(\frac{(2j+1)v\pi}{16}\right)$$

where i,j are the coordinates of a pixel in the basis image. Each macroblock $f(i,j)$ can be represented as a sum of these basis images:

$$f(i,j) = \frac{2}{N}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)B_{u,v}(i,j)$$

where:

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & x > 0 \end{cases}$$

The quantity $F(u,v)$ is an alternate representation of the block, and it represents the amount of each basis image in the actual block. This quantity is known as the discrete cosine transform (DCT) of the block. It can be obtained directly from the block $f(i,j)$:

$$F(u,v) = \frac{2}{N}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)B_{u,v}(i,j)$$

Why the transformation? Not surprisingly, most of the information in the blocks are present in the low frequency components (basis images with low u and v values). This means we can discard much of the information in the higher frequencies, and still present a good image.

To perform this elimination operation, the DCT block is quantized. This procedure is similar to what is done in speech compression. Each component of the DCT is mapped to one of a finite number of values. This mapping function is called a *quantizer*. Different quantizers are sometimes defined for different DCT coefficients. For example, the $u = 0$, $v = 0$ coefficient represents the average value of a block (plug these values into 1.2.2 to see this). This is often referred to as the DC component of the block, and is an important piece of information. As such, it is usually quantized very finely with more bits than the rest of the components. The highest frequency components are sent with very few bits at all, sometimes just one or two. The quantizer is usually controllable by the encoder. The allowed quantized values are often given by $MIN_VALUE, .. - 3Q, -2Q, -1Q, 0, 1Q, 2Q, 3Q, ...MAX_VALUE$ for a particular Q, settable by the encoder. Small values of Q (known as the *quantization parameter*) yield good reproductions of the original values, but require

more bits, since more quantized values are available. Higher values yield poorer reconstructions, but require fewer bits. This gives the encoder the ability to control the quality/bitrate tradeoff.

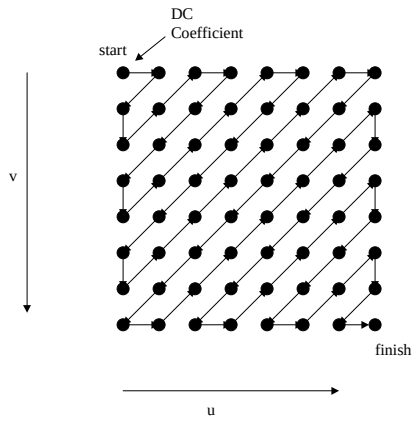

Figure 1.9: Zig-Zag Scanning of DCT Coefficients

After quantization, many of the coefficients will be zero (since zero is always one of the quantized values). To efficiently encode the block of coefficients, a procedure called run length coding is used. This procedure scans through the block in a zig-zag fashion (as shown in Figure 1.9), and the coefficients are written down in order as they are passed. The result is a sequence of numbers. The zig-zag process tends to put the numbers with lower frequencies towards the beginning of the list, and higher frequencies towards the end. Since the higher frequency components are often zero after quantization, this tends to place many zeroes towards the end of the list, and many non-zeroes towards the beginning.

This ordering allows a process known as *run length encoding* to be applied. In this process, a non-zero number, and all of the zero coefficients which follow it, are grouped together. The group is represented by a pair, representing the value of the non-zero number, and the number of zeroes which follow.

For example, the sequence:

```
-5, 0, 0, 1, 0, 0, 2, 3, -3, 0, 0
```

can be run-length coded as:

```
(-5,2), (1,2), (2,0), (3,0), (-3,2)
```

These pairs are then further compressed using a lossless procedure known as *Huffman coding*. The result is sent to the decoder. The decoder decodes the Huffman codes, resulting in run-length pairs. These are then converted to a sequence of 64 numbers, which are placed back into an 8x8 array. The inverse DCT is applied, and the resulting block is reconstructed. This process is applied to each of the six blocks in the macroblock.

Motion Compensation

The DCT takes advantage of *spatial redundancy*, that is, information that one pixel implies about the pixels next to it. However, there is also *temporal redundancy* in a video image. This means that a pixel in one frame is probably similar to a pixel in the previous frame. In fact, its quite likely that a macroblock in one frame is similar to a macroblock in the previous frame. In the absence of motion, a macroblock in one frame will be identical to the same macroblock in the previous frame. However, video has motion. A common type of motion is *translational motion*. In translational motion, objects move in a two dimensional fashion. This implies that a macroblock in a frame is similar to a 16x16 array of pixels somewhere in the previous frame, but perhaps at a different location. This is shown in Figure 1.10. Frame N contains a car and buildings in the background. In the next frame, $N+1$, the car has moved towards the right. The macroblocks containing the car exist in the previous frame, but translated horizontally.

The offset of the macroblock in the current frame, and its position in the previous frame is known as a motion vector. In the example of Figure 1.10, consider the macroblock three from the left, and six down, in frame $N+1$. This macroblock contains the top of the rear wheel. A near duplicate version of this macroblock exists in frame N, but approximately 10 pixels to the left and none up or down. As a result, we say that the *motion vector* for this macroblock is (-10,0).

The process of motion compensation takes advantage of this translational motion to reduce drastically the bitrate required to transmit a video sequence. Each macroblock is *predicted* from a previous frame by searching for block in the previous frame which is a good match. The result is a motion vector. Then, the difference between this predicted macroblock and the actual macroblock is computed. This yields a difference image. The DCT is applied to this image instead of the original image. Often this results in

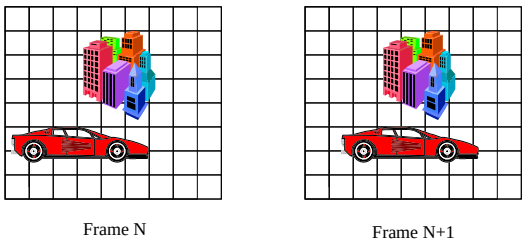

Figure 1.10: Translational Motion in Video

much less bits to represent than sending the macroblock itself. The DCT coded difference image, and the motion vector, are then sent to the decoder.

Note that the predicted block used by the encoder is not taken from the previous uncompressed frame, but from the previous compressed and decompressed frame. This is because the decoder only has access to the decoded frames. So, when it uses the motion vector to predict the macroblock in the current frame, it must use the previous decoded frame. Since the encoder sends the difference between the predicted macroblock and the actual macroblock, the encoder must also use the previous decoded frame.

Determining the right motion vector for each macroblock is a difficult task. This usual requires extensive searches, trying out various motion vectors until the best one is found. These searches are often the most complex part of the video encoding process. Since they are not done at the decoder, the video encoding process overall requires much more computation than the decoding process. For a particular codec (like H.263), different implementations will perform this motion search in different ways. A good search yields motion vectors that accurately predict the blocks, reducing the number of bits which must be transmitted. A poor search (or no search at all), yields bad estimates, increasing the bitrate. This allows for product differentiation of encoders even within the same specification.

Motion compensation is not the only search an encoder must do. Encoders must also decide whether each macroblock is best represented directly (without motion compensation), or by being predicted from the previous frame, and having the motion vector and difference block transmitted. A macroblock sent directly is known as an *intra* block, and a macroblock sent by means of a difference is an *inter* block. In some cases, the entire frame is sent using intra blocks. In this case, the frame is known as an I frame. Frames sent with some inter macroblocks are known as P frames, since portions of them are *P*redicted from the previous frame.

I frames and intra macroblocks are important for transmission over the Internet. Consider the case where the first frame of a video conference is sent using an I frame, and every macroblock in all subsequent frames were sent inter coded. If a packet is lost, some portion of the frame will be decoded improperly. Subsequent frames are predicted based on that frame, and so these too will be decoded improperly. The error will propagate, and eventually the decoded picture will be complete garbage, even as a result of a single bit error. To deal with this, the decoder must be periodically refreshed with a correct frame. By sending an I frame (or I macroblocks) frequently enough, this refresh can take place.

I frames are also important in stored video applications. If a user wishes to fast forward or rewind a video-on-demand movie, the ideal way to do so is to only transmit every other frame and play them at normal speed. If all frames were P frames, this would be impossible. The frame after the skipped one would depend on the previous. However, if every other frame is an I frame, fast forward and rewind at 2,4,6 or 8 (and higher) times the normal playback rate are possible.

Another type of frame commonly used is a B frame. A P frame is predicted from the previous frame. A B frame is bi-directionally predicted. This means it is based on macroblocks in the previous frame, and macroblocks in the next frame. This is shown in Figure 1.11. The figure shows four frames. The first is an I frame, the third and fourth are P frames, and the second is a B frame. An arrow from frame N to M means that N is predicted from data in M. Note that the B frame is predicted from the frame before, and the frame after. Like lookahead in speech compression, B frames given better compression rates at the cost of delays. The encoder must wait for the next frame before it can transmit a frame. At frame rates of 10 frames per second, this incurs a delay penalty of 100ms.

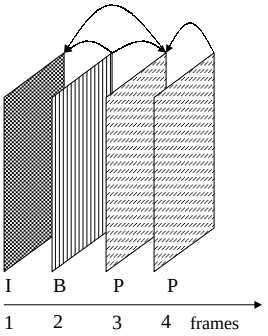

Figure 1.11: B Frames

1.2.3 Rate Control

An important facet of video compression systems for circuit switched networks was *rate control*. In most cases, there was a fixed amount of bandwidth available for transmitting the video. Unfortunately, unlike speech, video compression results in a bitstream with a variable bitrate. This means that sometimes the bitrate would be higher than the rate available, and sometimes lower. To keep the system functioning correctly, a feedback mechanism was deployed, as shown in Figure 1.12.

The system consists of the encoder, which generates the bitstream, and places it into a buffer. The buffer is drained at a fixed rate into the network. A control system monitors the occupancy of the buffer. If it gets too high, the control system increases the values of the quantization parameters used in the encoding process. This reduces picture quality, but decreases the bitrate. If the occupancy of the buffer gets too low, the quantization parameters are increased. In emergency situations, the system is able to drop entire frames so as not to overflow the buffer, or insert dummy bits (called *stuffing*) to ensure the buffer never underflows.

The need for such tight controls is eliminated in a packet network like the Internet. The amount of bandwidth available is not a fixed amount. It

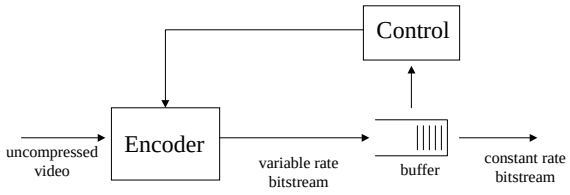

Figure 1.12: Rate Control Mechanism

depends on the quality of service provided by the network, on the congestion levels in the network, and the amount of bandwidth available along the path from sender to receiver. We discuss rate controls in this context further in chapter XXX.

Complete System

Motion compensation, discrete cosine transform, quantization, and rate control form the integral components of a video compression system. A system using these components is depicted in Figure 1.13. The video enters the system. If the block is to be intra coded (as determined by the I/P decision box), the frame is passed into the DCT component. This is followed by a quantization (Q) component. The data is then encoded (using run length codes and Huffman codes), and sent to the decoder. The inverse quantization (Q^{-1}) followed by inverse DCT (DCT^{-1}) reconstructs the frame as it will be reconstructed at the decoder. This frame is then stored, and can be used for predicting the next frame.

If the I/P device decides to inter-code the frame (a P frame), the switches are thrown, and the input video is subtracted from the motion compensated (MC) version of the previous frame. The compensation (i.e., prediction)

is based on a motion estimation (ME) block that determines the motion vectors to use. The subtracted frame is then passed through a DCT and quantizer, and sent to the decoder. The inverse process is applied, and the difference frame is added to the previous decoded frame, and stored.

Finally, a control component monitors the encoding process, adjusting the quantization process to perform rate control.

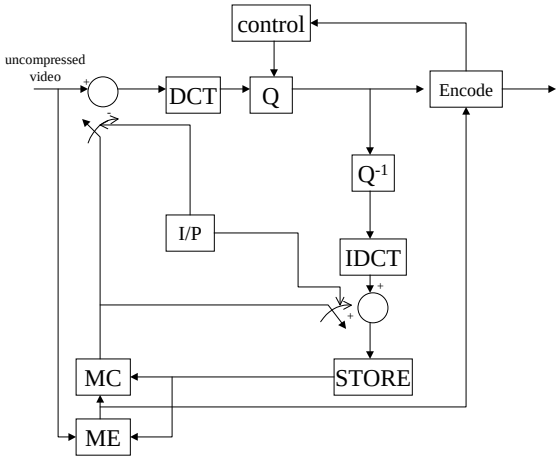

Figure 1.13: Video Encoding System

1.2.4 Compression Standards

A number of standards have been developed for video compression. These standards specify the formatting of bitstreams that an encoder must generate, and the process of decoding. The way in which the encoding is done - choosing quantization parameters, performing motion compensation, and choosing between I, P, and B frames, is left to the discretion of the implementor. This allows implementations to differentiate each other, and yet still be interoperable.

In this section, we briefly review the common standards for video compression on the Internet.

H.261

H.261 is specified by the ITU, and is formally named "Video Coding for Audio-Visual Services at Px64 kb/s". It was published in March of 1993 [10]. Its initial application was for video-conferencing over Integrated Services Digital Networks (ISDN). ISDN provides circuit switched channels in multiples of 64 kb/s. The aim was to develop a video conferencing codec which could operate well with some number, P, of these channels.

H.261 operates on input video frames in two sizes - the Common Interchange Format (CIF), and the Quarter CIF (QCIF). CIF frames have 352 pixels horizontally, and 288 pixels vertically. QCIF frames have 176 pixels horizontally, and 144 pixels vertically. Each pixel has 24 bits of color, represented using a luminance (Y) and two color difference components (C_r and C_b).

H.261 allows for motion compensation (using motion vectors from -15 to +15 in each direction), DCT, run length coding, Huffman coding, and quantization, as described in the previous section. It also defines a filter (called a loop filter) which can be applied to the predicted macroblocks before subtracting them from the input frame to obtain the difference.

H.261 also further divides the picture into Groups of Blocks (GOB). Each GOB contains 11 macroblocks horizontally, and 3 macroblocks vertically. In a CIF picture, there are 13 GOBs, and 3 in a QCIF picture. When the bitstream is sent to the decoder, a *start code* is inserted at the beginning of each GOB. The start code is a special sequence of 16 bits (0000 0000 0000 0001) which can never appear anywhere else in the bitstream. Should the decoder lose synchronization, it can *hunt* for the start code by searching the bitstream until it finds it. At that point, it knows it has found the beginning of a GOB, and decoding can start from there. The use of these synchronization markers is important when video is transported over IP. Packet losses cause loss of synchronization, so that the decoder must hunt for the start code once more. See section XXX for a more detailed discussion.

H.263

ITU Recommendation H.263, entitled "Video coding for low bitrate communication", builds on H.261 by extending it with new picture sizes, features, and options. The basic encoding principles (DCT, quantization, Huffman coding, etc.) are the same as H.261.

H.261 defines the CIF and QCIF formats, which are inherited in H.263. However, H.263 also adds sub-QCIF (128x96), 4CIF (704x576), and 16CIF

(1408x1152), and allows for the definition of additional custom formats.

H.263 adds a number of optional features. An encoder can use these features if it is certain the decoder understands them. Determining this is the function of capabilities exchange, and is handled by other protocols, such as H.245 or SDP.

Some of the new features include:

Unrestricted Motion Vectors In normal operation, a motion vector cannot cause a macroblock to be predicted from pixels outside the picture. For example, the rightmost macroblock in a frame can't be predicted using motion vectors with positive horizontal components. H.263 allows this restriction to be lifted. When a macroblock is predicted in this way, the missing pixels are extrapolated by copying the edge pixels outside the picture. This feature is useful when there is motion across the border of the picture.

Syntax-based Arithmetic Coding Instead of using Huffman coding, an alternate representation, called Syntax-based Arithmetic Coding (SAC), can be used for DCT coefficients. This encoding uses fewer bits, but is more complex to encode.

Advanced Prediction Mode Normally, only a single motion vector for a 16x16 macroblock is transmitted. In advanced prediction mode, four motion vectors can be transmitted, one for each 8x8 luminance block. In addition, the actual predicted 8x8 block is a weighted sum of three predicted blocks, each using a different motion vector. One is the predicted block using the motion vector of the 8x8 block itself, and the other two are based on using motion vectors from surrounding blocks. This is known as *overlapped motion compensation*. Advanced prediction mode provides better picture quality, at the expense of increased encoding complexity.

PB Frames H.263 supports the concept of a PB frame. A PB frame are actually two pictures coded as a single unit. One picture is a P frame (predicted from the previous I or P picture), and the other is a B frame, predicted from the P frame piggybacked with it and the previous I or P frame. By coding the two as a single unit, greater compression can be achieved by exploiting redundant information between the two. H.263 allows each macroblock in the B component to be predicted from the previous picture (forward prediction), from the P component of the PB frame (backwards prediction), or a combination of both (bi-directional prediction).

Advanced INTRA Coding Mode H.263 allows for an advanced INTRA coding mode. Normally, the coefficients for an INTRA block represent the DCT of the block itself. The coefficients for an INTER block represent the difference between a predicted value from the previous frame and the current frame. An advanced INTRA coded block is a hybrid between the two. Some of the coefficients for the current block are predicted from a neighboring block in *the same* picture. The difference between these predicted values and the actual coefficients are then sent. Not all DCT coefficients are predicted. H.263 allows (1) just the DC coefficient to be predicted, (2) the coefficients with horizontal components only to be predicted, or (3) the coefficients with vertical components only to be predicted. The prediction is done from either the block above or to the left (or both), depending on which coefficients are being predicted. Advanced INTRA mode provides better coding efficiency for many INTRA blocks.

Deblocking Filter Mode A filter can be added to the decoded picture before it is stored for reference for predicting the next frame. This filter causes the edges between 8x8 blocks to be smoothed, resulting in improved picture quality.

Slice Structured Mode In H.261, the picture was divided into a fixed number of GOB's. These GOB's provided resynchronization points. When a packet is lost, the decoder can find the beginning of a GOB and start decoding from there. In H.263, this feature is enhanced. Instead of GOB's, the picture is divided into slices. Each slice contains some number of macroblocks. The slices are rectangular in shape, and can contain arbitrary non-overlapping sections of the picture. The slices can actually be transmitted in arbitrary order. Each slice is an independent unit. The slice can be defined so that motion vectors cannot predict a macroblock from pieces of a picture outside of the slice, and advanced intra mode cannot be used to predict a block from blocks outside of the slice. In this way, when a packet is lost, only the slices with data inside the packet are affected.

Reference Picture Selection Mode Normally, a P frame is predicted from the previous P (or P component of a PB) frame. In reference picture selection mode, a P frame can be predicted from a P frame a certain number of frames in the past. Furthermore, H.263 allows a back channel from the decoder to the encoder, indicating which frames have been received correctly. This can be used by the encoder

to predict the current frame from a frame which it knows the decoder received correctly. The result is a limitation in the propagation of errors due to packet loss. This mode comes at the cost of increased memory requirements at both encoder and decoder (they must store a number of frames from the past), in addition to encoding complexity. However, it is very useful for Internet telephony because of its error limiting effects.

Scalability Modes *Scalability* refers to the ability to encode the bitstream in layers, such that it is only necessary to receive a single layer to decode the picture, but additional layers provide improved picture quality. Scalability is often used with multicast distribution, where each layer is sent on a different multicast group. We discuss this further in Section XXX. H.263 provides three scalability modes - temporal, SNR, and spatial. In temporal mode, the extra layers allow the frame rate to be improved. This is done by means of B frames. B frames can be used for this purpose since no other frame is ever predicted from them. They are therefore not necessary at the base layer to decode. SNR scalability means that additional data is sent to improve the quality of each picture. This data is sent as an enhancement frame. In spatial scalability, receivers that receive the higher layers first increase the resolution of the decoded pictures by interoplation, and then add an enhancement picture to improve sharpness.

Reduced Resolution Update The Reduced-Resolution Update mode is expected to be used when encoding a highly active scene, and provides the opportunity to increase the coding picture rate while maintaining sufficient subjective quality. This mode allows the encoder to send update information for a picture that is encoded at a reduced resolution, while preserving the detail in a higher resolution reference image to create a final image at the higher resolution.

Bibliography

[1] W. Kleijn and K. Paliwal, *Speech Coding and Synthesis.* Amsterdam: Elsevier Press, 1995.

[2] International Telecommunication Union, "Pulse code modulation (pcm) of voice frequencies," Recommendation G.711, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Nov. 1998.

[3] International Telecommunication Union, "40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (adpcm)," Recommendation G.726, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Dec. 1990.

[4] International Telecommunication Union, "Coding of speech at 16 kbit/s using low-delay code excited linear prediction," Recommendation G.728, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Sept. 1992.

[5] International Telecommunication Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction," Recommendation G.729, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1996.

[6] International Telecommunication Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction annex a: Reduced complexity 8 kbit/s cs-acelp speech codec," Recommendation G.729A, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Nov. 1996.

[7] International Telecommunication Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction annex b: A silence compression scheme for g.729 optimized for terminals conforming to recommendation v.70," Recommendation G.729B,

Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Nov. 1996.

[8] International Telecommunication Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction annex d: 6.4 kbit/s cs-acelp speech coding algorithm," Recommendation G.729D, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Sept. 1998.

[9] International Telecommunication Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction annex e: 11.8 kbit/s cs-acelp speech coding algorithm," Recommendation G.729E, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Sept. 1998.

[10] International Telecommunication Union, "Video codec for audiovisual services at px64 kbit/s," Recommendation H.261, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1993.