Multimedia Systems Manuscript No.
(will be inserted by hand later)

Silence detection for multimedia communication systems

Stephen Jacobs, Alexandros Eleftheriadis, and Dimitris Anastassiou

Department of Electrical Engineering
Columbia University, New York, NY 10027, USA

April 13, 1997

Abstract. Silence detection and removal is an essential building block of any multimedia video conferencing system. It reduces the bandwidth requirements of the underlying network transport service and helps to maintain an acceptable end-to-end delay for audio. We analyze the requirements for a silence detection algorithm hosted on a multimedia communication system, and propose a novel low complexity algorithm operating in the non-linear μ-law domain. After discussing the constraints which are imposed by the architecture of the system hardware (computer, packet-based network), we show that several recently proposed silence detection algorithms fail to meet all of these constraints. A new approach is then introduced, based on the small- and large-signal behavior of the speech waveform in the μ-law domain. The new algorithm is compared with a recent design that meets several of our requirements; experimental results indicate that it performs significantly better in the particular environment at hand.

Key words: Multimedia communication systems – Silence detection – Video conferencing algorithms

1 Introduction

As a result of recent advances in video compression, processor design, and network architecture, it is now quite feasible to implement multimedia communication applications (e.g. video conferencing) using standard computing and networking facilities (LeGall 1991; Eleftheriadis et al. 1993; Eleftheriadis et al. 1994). This shift of multimedia communication equipment and services from dedicated systems to general purpose computers and packet-based communication networks has introduced a quite different operating environment and has prompted the reexamination of several key algorithms.

Although compression is undoubtably a crucial component in the designer's arsenal, recent work (Eleftheriadis et al. 1994, and references therein) has shown that the flexibility provided by a general-purpose platform introduces challenges that did not exist in monolithic, dedicated designs. The primary source of these challenges is the Quality of Service (QoS) provided by both the network transport layer and the host operating system layer. The communication path cannot be modeled as a bit-pipe with a constant transmission delay as in circuit-switched connections. Also, sophisticated algorithms are required in order to maintain QoS. As an example, the large delay variation (jitter) that may be present in networks with limited or no QoS guarantees can have severe effects on the end-to-end delay. Similarly, the capability of guaranteed quality transmission in Asynchronous Transfer Mode (ATM) and other networks presupposes the existence of an appropriate resource reservation and admission control mechanism.

In this paper we are concerned with the design and implementation of a silence detection and removal algorithm in such multimedia communication environments. The purpose of this algorithm is to identify and remove long runs of silence from the audio signal stream. Its importance as a building block stems from two basic problems. First, the bandwidth requirements of video are significant, and any reduction in terms of the total bandwidth required by the system is an important benefit. We should note that the bandwidth requirements for high-quality audio in a multimedia system can be quite close to that of low-quality video, and hence this may represent a significant fraction of the total bandwidth. For example, in the MBONE (the multicast enabled part of the internet) video is transmitted using H.261 at a rate of 128 kbps to 256 kbps, which is on the same order of magnitude as 64 kbps audio. Of course, silence detection and removal cannot solve all of the problems in multimedia communications; no amount of silence detection can help if there is simply not enough bandwidth in the network to support audio transmission.

Second, and more importantly, the removal of silence helps to reduce the end-to-end delay seen by the users (Eleftheriadis et al. 1993). This is because it transforms the constant bit rate source to a bursty one which can

be accommodated more easily in networks with limited QoS guarantees.

The structure of the paper is as follows. In Sect. 2 we discuss in detail the particular features of a multimedia communication environment, and derive a set of design requirements for an appropriate silence detection algorithm. These are contrasted, in Sect. 3, with several recently proposed silence detection schemes, and it is shown that none meets all of our design criteria. In Sect. 4 we describe a novel, low-complexity silence detection algorithm that operates directly in the non-linear μ-law domain. The algorithm capitalizes on the small- and large-signal behavior of the μ-law companded speech waveform; due to its low complexity, it can be easily implemented in software and can be directly incorporated into existing multimedia communication system designs. In Sect. 5 we present experimental results that verify the effectiveness of the proposed algorithm. We also perform a comparative analysis with a recently proposed silence detection scheme that meets several of our design criteria, and show that our approach outperforms it in several key areas. Finally, in Sect. 6 we present some concluding remarks.

2 Silence detection in multimedia communication environments

The generality of the concept of a multimedia communication system makes it difficult to capture it in a universally acceptable definition. For our purposes, such a system is assumed to be composed of a general purpose computer and a computer network. The most important characteristic of such an environment is that all data transfer and communication is handled in packets. This includes acquisition (delivery of data from the audio device to the operating system), application handling (transfer of data by the application between the audio device and the transport layer), and network delivery (actual transfer of data over the network) at all protocol layers. A typical example would be a set of workstations connected via an Ethernet or ATM local area network.

An important parameter of such multimedia communication environments is the QoS support they can provide. In contrast with data-oriented communication, video and audio impose strict requirements on available bandwidth and end-to-end delay. These requirements affect all system components, from the sender to the receiver. High-quality communication can only be provided if the total system behavior satisfies those requirements. At the lowest end, a purely data-oriented system provides no QoS guarantees: throughput and delay depend on the current state of the hosts and the network they reside on. At the highest end, both bandwidth and delay are assured within prespecified and agreed-upon bounds. Cases in between, where bounds are provided for delay only, are also possible.

There is a direct tradeoff between bandwidth availability and end-to-end delay. To illustrate this, consider the example of a constant bit rate audio signal being transported over a multi-access network such as Ethernet. The signal is buffered at the receiver, using a buffer size of B samples, and playback starts after a prespecified buffer occupancy B_{w} is reached. The end-to-end delay D is determined by the time needed to obtain the audio data (acquisition delay) τ_{a}, the transmission delay τ_{t}, and the waiting time in the receiver's output buffer τ_{w}. The acquisition delay is a result of the frame-based structure of the sampling process; the audio samples are placed into a buffer by the audio device driver and delivered to the application by the operating system when the buffer has filled. We then have:

$$D = \tau_{\mathrm{a}} + \tau_{\mathrm{t}} + \tau_{\mathrm{w}} \ . \tag{1}$$

If the end-to-end delay of the network were constant, then the buffer occupancy would remain constant on the average since both the source and receiver produce and consume audio samples at the same rate, ignoring any possible clock mismatch. Denoting the sampling rate of the audio signal as r, in steady state we would then have:

$$D = \frac{B_{\mathrm{a}}}{r} + \tau_{\mathrm{t}} + \frac{B_{\mathrm{w}}}{r} \ , \tag{2}$$

where B_{a} is the acquisition buffer size.

In case significant congestion occurs in the network, audio frames at the source will start accumulating, awaiting transmission, which increases τ_{t}. After congestion subsides, these frames will be transmitted to the receiver, and upon arrival will encounter an empty buffer if the duration of the congestion is long enough. Denoting by τ_{c} the congestion duration, receiver buffer starvation will occur if:

$$\tau_{\mathrm{c}} \geq B_{\mathrm{w}}/r \ . \tag{3}$$

Since audio samples cannot be removed from the receiver buffer faster than their playback rate r, the end-to-end delay increases by the amount of time the receiver buffer remains empty, $\Delta\tau = \tau_{\mathrm{c}} - B_{\mathrm{w}}/r$. The new end-to-end delay, in the case of receiver buffer starvation is:

$$D' = \frac{B_{\mathrm{a}}}{r} + \tau_{\mathrm{t}} + \frac{B_{\mathrm{w}}}{r} + \Delta\tau = \frac{B_{\mathrm{a}}}{r} + \tau_{\mathrm{t}} + \tau_{\mathrm{c}} \ . \tag{4}$$

Note that this increased end-to-end delay will be present for the remainder of the session, or even possibly increased if congestion periods longer than τ_{c} occur in the future. Each time silence is removed, τ_{c} is decreased by B_{s}/r, where B_{s} is the amount of silent data removed.

The output buffer at the receiver can be thought of as an M/D/1 queue, which has random arrivals and fixed length packets. The audio data arrives at the queue at a rate of r samples/sec and is played back at the same rate. Therefore the offered load to the queue is 1. This means that the queue will eventually fill. In practice, the buffer is of finite size and the probability that it will overflow due to jitter can be reduced by making it large enough. Such an approach is typically unacceptable, however, since it would incur a very high end-to-end delay.

In multi-access networks jitter can be extremely variable; experiments (Eleftheriadis et al. 1994) have shown

that the end-to-end delay on a moderately loaded Ethernet can reach 1 sec in less than 1 min. Long distance telephony standards prescribe end-to-end delays as being acceptable if they are below 200 msec. Even in applications not involving person-to-person communication, such as video-on-demand, end-to-end delay is still important because it has a direct impact on interactive response time.

Since it is not possible to control the transmission delay in this type of network, all other controllable delays must be minimized; this includes the acquisition/processing time and the buffer waiting time. One of the purposes of silence detection and removal is to reduce the arrival rate so that the offered load is less than 1. This is accomplished by exploiting the fact that during an average conversation, each subscriber speaks only 40%-50% of the time (Drago et al. 1978). If no data were sent the other 50%-60% of the time, the arrival rate would be reduced to about $0.5r$ samples/sec and the offered load would be reduced to about 0.5. The expected number of packets waiting in an M/D/1 queue is (Schwartz 1987):

$$E(n) = \frac{\rho}{(1-\rho)}(1 - \frac{\rho}{2}) \ . \tag{5}$$

With $\rho = \frac{1}{2}$, $E(n) = \frac{3}{4}$ packets. Thus, the output buffers would not fill and on average there would only be about $\frac{3}{4}$ packet in queue.

Silence detection and removal for a video conferencing (or general multimedia communication) system is not concerned with inter-syllabic or inter-word silences. In fact, removal of these silences would degrade the quality of speech due to the temporal non-linearity. It would create an effect of talking very quickly since the syllables and words would run very closely to each other. Studies have shown that 99.56% of "continuous speech" segments have periods of silence smaller than 150 msec (Lynch et al. 1987). For this reason, and using a 64 Kbps (8 KBps) signal as an example, removal of a block of silence smaller than 1200 bytes would reduce the quality of speech. Also, the performance improvement gained by removing inter-word and inter-syllabic silence would be quite small in comparison to the substantial removal of 50%-60% of the signal, which will not affect the perceived quality of the audio.

Summarizing the above discussion, a silence detection algorithm for this particular environment should then meet the following five requirements:

1. It should remove as much silence as possible (in lengths of 150 msec and more) and should not affect the quality of speech;
2. it should have fairly low complexity since software implementations are desirable, and also because the processing delay will be added to the end-to-end delay;
3. it should require a small buffer since it is operating in an environment with limited resources;
4. it should be able to make a speech/silence decision within a frame, since storage of a frame would increase the end-to-end delay by the frame's duration; and
5. it should detect long runs of silence rather than short ones.

An additional reason in support of the last requirement is that when silence is removed from a frame, the following samples in the frame must be moved back to the point where silence occurred in order to create a continuous—yet shorter—packet. If this happens several times within a frame, the processing time for the memory-to-memory copies can become significant. The argument still holds even if scatter-gather I/O is supported by the transport layer routines.

3 Silence detection algorithms

Since the issue of silence detection has been explored time and again since the Time Assigned Speech Interpolation (TASI) plan in 1959 (Bullington and Fraser 1959), a large variety of research is available which approaches the problem from a broad spectrum of viewpoints. In its simplest form it can be a magnitude based decision. The algorithm compares the magnitude of the signal against a preset threshold. If a percentage of the data is smaller than the threshold, silence is declared. This algorithm has fairly mediocre performance in the presence of any background noise. It does however, meet the other requirements for low complexity, low buffer size, large run length, and intra-frame decision. More sophisticated approaches are based in differential decision logic (Eleftheriadis et al. 1993). Here the speech/silence decision is based on the difference between successive samples. The heuristic behind this is that speech has larger magnitude variations than silence or even moderate background noise.

Recent research has increased performance but at the added cost of complexity. Rangoussi, et al., approached the problem from the mobile communications perspective where very low signal to noise ratios exist. Their work required a robust, even if computationally more demanding, method. They used third-order statistics by exploiting the non-linearity of speech to accurately endpoint the silence/speech changeovers (Rangoussi et al. 1993). Un and Lee developed a technique using linear delta modulation bit streams (Un and Lee 1980). DonVito and Schoenherr used the advantages of subband coding to aid in their attempts to perform silence detection (DonVito and Schoenherr 1985). Other currently used methods are zero-crossing rates, signal energy, one sample delay correlation coefficient and prediction error energy (Savoji 1989). The difficulty with all of these is that they were not designed to tackle the specifics of the presented problem and are also too complex, particularly for real-time software-based implementations on general purpose computers.

An algorithm proposed by Savoji (Savoji 1989), enhancing an algorithm originally proposed by Lynch (Lynch et al. 1987), satisfies several of our criteria. The modified algorithm has the lowest complexity of the available research and still performs quite well. It creates and maintains adaptive metrics for speech and noise and then uses

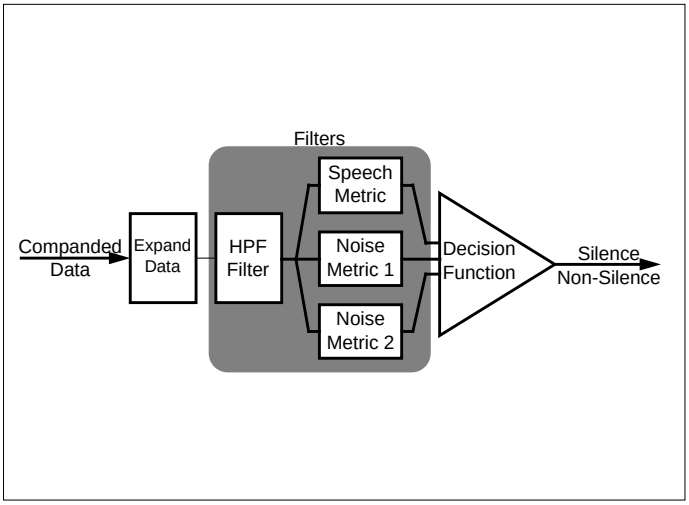


Fig. 1. Block diagram of Savoji algorithm

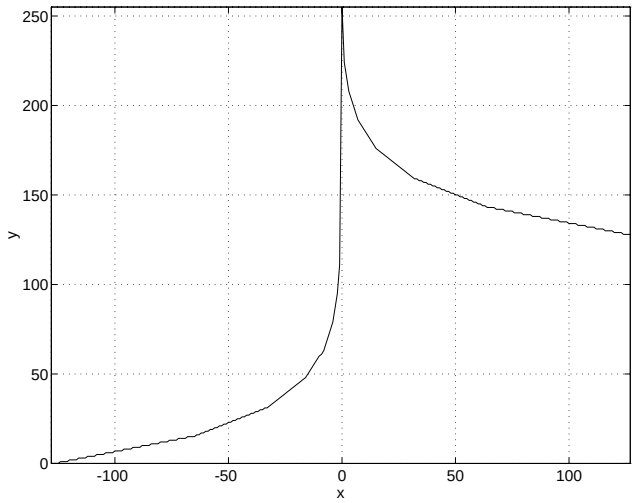


Fig. 2. μ-law function

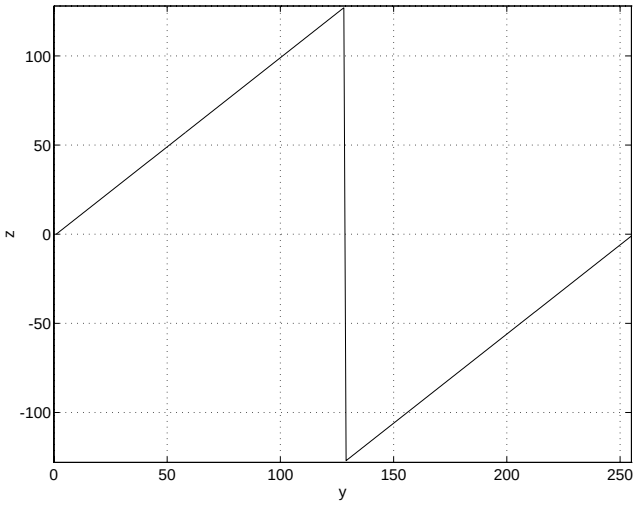


Fig. 3. 2's complement function

them, coupled with the short term energy, to make silence/speech decisions. The algorithm first transforms the data by converting from μ-law and then high-pass filters it, as shown in Fig. 1. Then the metrics are established by filtering the transformed data. The resulting metrics, coupled with a simple short term energy calculation, form the basis for a speech/silence decision.

Although its complexity is low compared with the other schemes, it uses 4 filters that each require 1 previous sample. It is shown in Sect. 5 that this approach meets most, but not all, of the five requirements. This technique is used for comparison purposes with the algorithm proposed in this paper.

4 Non-linear silence detection in the μ-law domain

When obtaining an audio frame, the input device collects the data and then μ-law compands it. Companding a signal *com*presses it at the source and then ex*pands* it at the receiver. In all of the previously mentioned work, data are decoded to their linear form, with the exception of Drago (Drago et al. 1978). The authors there use an approximation by assuming linearity at low signal levels. Since this conversion consumes valuable CPU time, our proposed algorithm exploits the non-linearity of the μ-law rather than trying to sidestep it.

The μ-law compander is used to give finer quantization to low signal levels than would ordinarily be obtained through linear quantization. If the input signal is x and the output y, then the formula for the μ-law transformation is

$$y = \frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \quad \text{for } -1 \le x \le 1\,, \tag{6}$$

where $\mu = 255$ for the United States digital carrier system (Bell Labs 1982) and $0 \le y \le 1$. This formula is shown in Fig. 2 for the case where x is an 8-bit signed byte and y is interpreted as an 8-bit unsigned byte.

After the μ-law transformation, the result is interpreted in 2's complement. If y is as above and is input to the 2's complement transform, then the output z is given by:

$$z = \begin{cases} y & \text{for } 0 \le y \le 127 \\ y - 256 & \text{for } 128 \le y \le 255 \end{cases}\,, \tag{7}$$

(see Fig. 3). The composite of these two functions is shown in Fig. 4 as a function mapping x to z. The value z will be referred to as the *Magnitude Factor* (MF).

An assumption that can be seen quite clearly from empirical data is that voice has no DC component, i.e. it is a zero-mean signal. To demonstrate the use of the MF, a simple example is needed. Referring to Fig. 4, if a small magnitude, zero mean signal with frequency π is the input to the MF function, the positive peak of the voice signal will create a small negative MF, while the negative peak of the signal will create a large positive MF. If the output of the MF function is then averaged over time, or low-pass filtered, the result will be positive. The low-pass filtered MF signal will be called the *Average Magnitude Factor* (AMF).

The same analysis can be performed on a large signal by again referring to Fig. 4. The positive peak of the

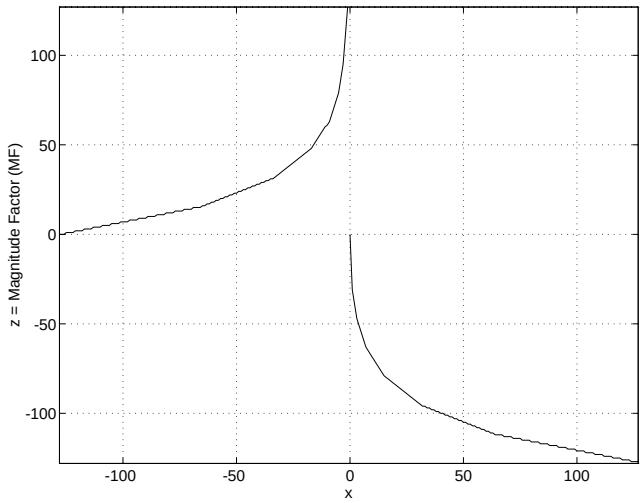


Fig. 4. Composite function

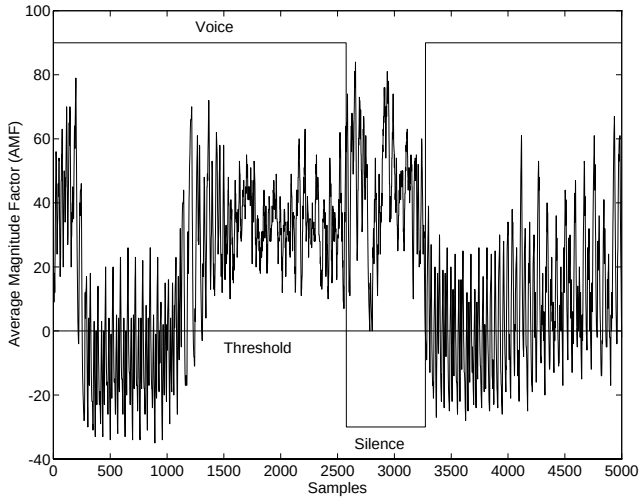


Fig. 5. AMF for a voice/silence/voice segment

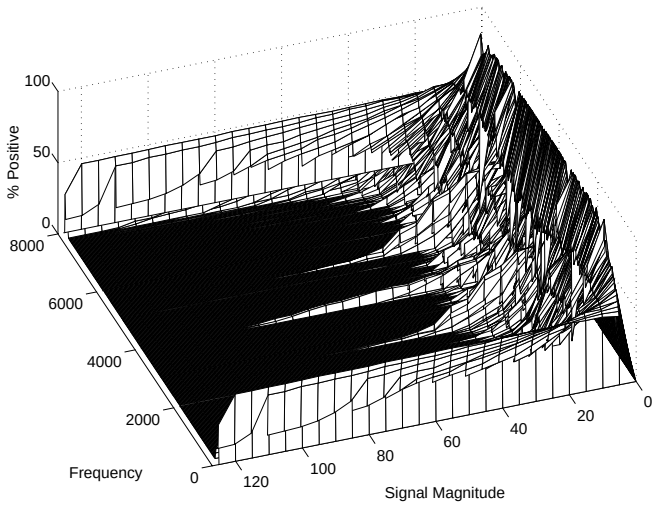


Fig. 6. Percentages of positive AMF samples for varying frequency and magnitude

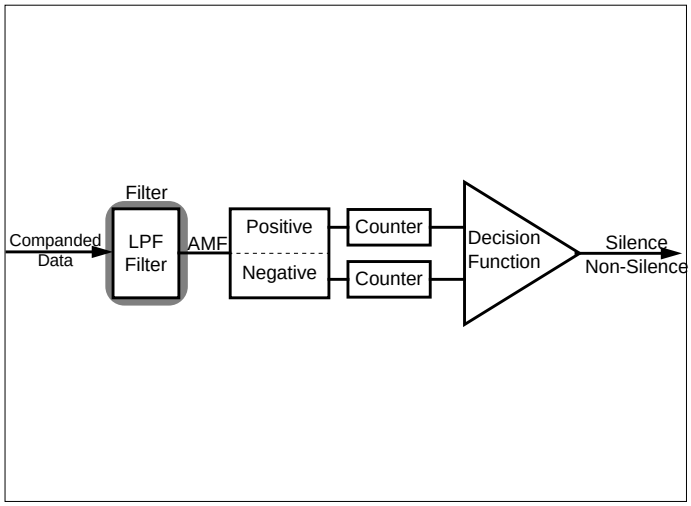


Fig. 7. Block diagram of proposed algorithm

voice signal will generate a large negative MF while the negative peak will create a small positive MF. The AMF will therefore be negative. An example of the AMF is shown in Fig. 5, where the input is a segment which contains voice and silence. As can be seen the AMF is much greater than zero only between 2500 and 3500 samples. This is exactly the segment corresponding to silence. Superimposed on this signal is a square wave demonstrating where the algorithm actually detected the silence based on the AMF.

The behavior of this non-linear transformation for sine waves of various frequencies and magnitudes is depicted in Fig. 6. We see that at small magnitudes we obtain a larger portion of positive samples than negative ones. At larger magnitudes, the transformation yields primarily negative samples. This corresponds to the low valley in the graph. The small and large signal analysis in the previous paragraphs assumed signals of frequency π, which in this case, corresponds to 4000 Hz. This assumption was made for ease of analysis. A cosine wave at this frequency would have values of ± 1 which would correspond to the positive and negative peaks referred to above. These peaks pick off the various MF's, which are then averaged to obtain the AMF. A decision is then made based on the percentage of AMF samples that are positive.

Since the transformation is non-linear, superposition does not hold. For this reason, we cannot generalize this discussion to that of arbitrary signals. We present this line of reasoning here merely to foster the intuitive notion of why it works, and defer actual experimental proof to the results presented in Sect. 5.

So far the averaging required to create the AMF has been referred to as a filter. Very simple filters do not provide the smoothing necessary to provide an AMF that is stable over small blocks of time. After extensive experimentation, we found that a Chebyshev first order filter with cutoff frequency at 50 Hz yielded very good results (Oppenheim and Schafer 1989). The Chebyshev filter attenuates the higher frequencies enough to obtain a more slowly varying AMF so that the appropriate decision can later be made.

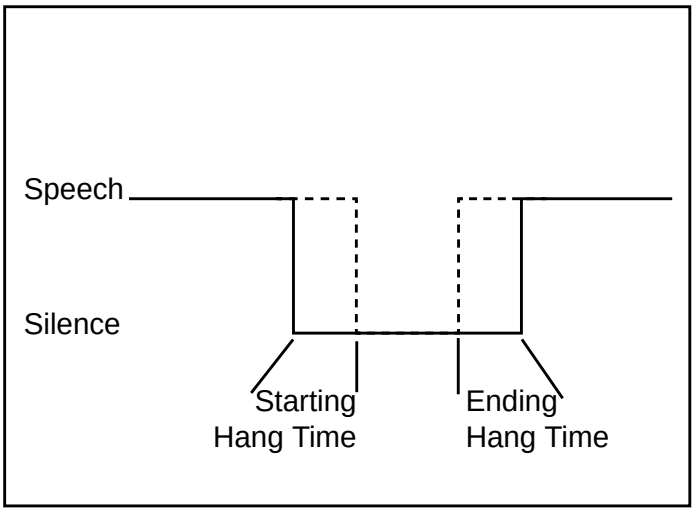


Fig. 8. Definition of starting and ending hang time

A block diagram of the algorithm is shown in Fig. 7. In our experiments, audio is acquired at 64 Kbps, using 8-bit μ-law companded samples (8 KHz sampling frequency). The A/D converter provides a 1024 sample frame through the operating system's device driver. Although, smaller frames would yield a lower end-to-end delay, the particular audio device that was being used had a lower limit of 1024 bytes. The frame is then examined by breaking it into smaller blocks of 256 bytes. Each block is determined to be silence if fewer than 25% of the AMF samples in the block are negative; otherwise the block is deemed speech.

A commonly employed technique for silence detection and removal algorithms is to use a starting and ending hang time. The starting hang time is the time between when silence is detected and when data is actually removed. Figure 8 shows where silence and speech were detected using a solid line. The dashed line represents where silence was actually removed based on the hang times. The purpose of the starting hang time is to make sure that the removal of silence doesn't inadvertently remove the end of a talk spurt.

The ending hang time is the time between when the algorithm stops removing data and the beginning of a talk spurt. Its purpose is to ensure that the beginning of a talk spurt is not removed. It is important to note that the hang times do not contribute in any way to the end-to-end delay.

The starting hang time was set to seven consecutive blocks or 224 msec. This was determined experimentally because shorter hang times resulted in truncation of the end of talk spurts. Once the starting hang time is reached, all subsequent and consecutive silent blocks are removed.

If a non-silent block is detected within a frame, no data will be removed from that frame. This corresponds to a variable length ending hang time of between 96 msec and 0 msec. For example, if silence has been detected and removed in previous frames, but the last block of the current frame is non-silent, no silence will be removed from the current frame. This corresponds to an ending hang time of 96 msec because the algorithm has detected the other three blocks of silence and is not removing them.

Since the ending hang time is variable, it can also be 0 msec. If silence has been detected and removed in previous frames and the current frame has all silent blocks, then this entire frame will be removed. If the first block of the subsequent frame is non-silent, then none of that frame will be removed and the ending hang time will be 0 msec. In theory, this large range is not ideal but is the best that can be done in a system where frames cannot be delayed by buffering. In practice, it only slightly degrades removal performance while protecting, above all else, quality of speech.

The determination to remove silence happens only at the end of a frame. This means that the removal is accomplished by sending only the part of the frame that corresponds to voice. Since the silence will always be at the end of the frame, data is never moved; the frame is merely truncated. This significantly decreases the complexity of the algorithm in comparison to others. It is also less complex because the conversion from μ-law to linear is eliminated and only one filter is used.

The algorithm meets the other requirements, too. Only one previous sample is needed across frames, hence satisfying the low buffer size specification. Since the silence/speech decision is made at the end of each frame, previous frames are not stored. Finally, the average period of silence detected is quite large and the quality of speech is completely unaffected. In short, all of the requirements are met and the last analysis to be made is how much silence is actually removed.

5 Performance results

In analyzing the performance of the proposed algorithm, we compared it to the technique demonstrated by Savoji (Savoji 1989) since it satisfies several of our requirements for a scheme appropriate for multimedia systems. In both techniques, there was no degradation in the quality of speech. The Savoji technique initially removed much more silence without cutting off words. However, the silence it was removing was inter-word and inter-syllabic silence. To alleviate this, the constraint was imposed that only periods of silence longer than 125 ms were to be removed. The result of modifying the algorithm to fit this situation was that the amount of silence removed was dramatically decreased, and in general was smaller than the silence removed by the proposed algorithm. Experimental results are shown in Table 1.

The 64 Kbps, μ-law companded audio signals for these experiments were obtained under the typical conditions for a video conferencing session. They were recorded at a workstation in a laboratory with a variety of different background noise sources such as computer fans and cooling systems. Silence was recorded with a large range of noise intensities, as indicated by the first column in Table 1.

The most important test cases are those that have extended periods of silence with and without background

Table 1. Comparison of Savoji and proposed algorithms in kilobytes

Type of silence	Percent removed proposed	Percent removed Savoji	Percent more removed by proposed	Average length proposed	Average length Savoji
Almost no noise	91	73	18	15.7	1.6
Low noise	85	64	21	14.8	1.5
Moderate noise	69	0	69	6.5	0.0
Loud noise	0	0	0	0.0	0.0
Inter-sentence silence	46	0	46	3.5	0.0

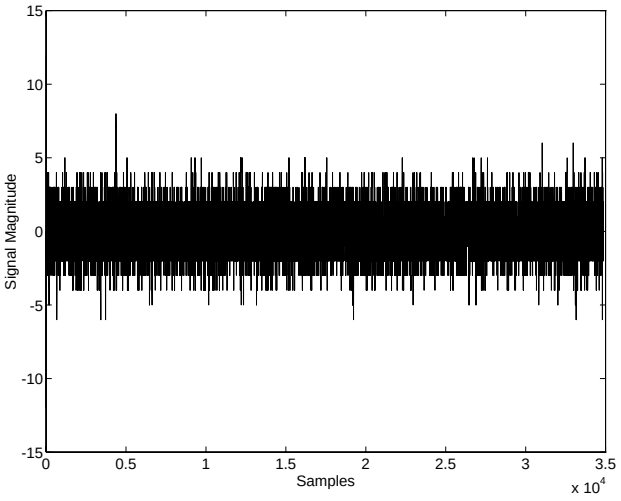


Fig. 9. Silence with low background noise

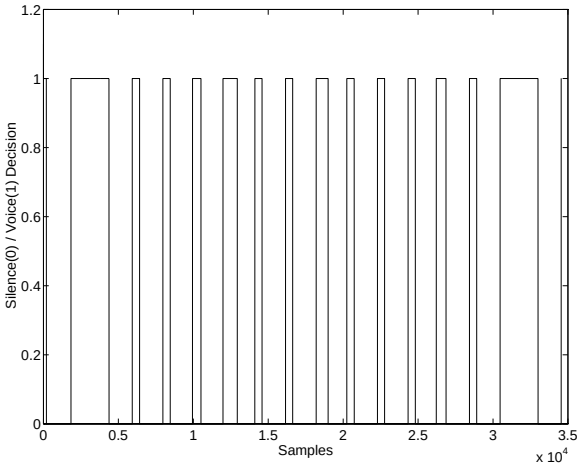


Fig. 10. Savoji algorithm operating on low noise signal

noise, i.e. the first four in the table. It is evident that at this level of complexity, very loud background noise could not be detected and removed. However, the results for moderate background noise, corresponding to nearby air conditioning system and computer fans, proved favorable. They showed that the proposed algorithm removed over 2/3 of the signal while the modified Savoji technique could not detect silent periods that were large enough to merit any removal at all. Low background noise had even better removal percentages.

Also demonstrated in Table 1 is the fact that, in general, the average length of silence removed is much higher for the proposed algorithm. Even though the proposed algorithm has no constraint for removing a minimum of 125 ms or 1000 bytes, the average length removed was much higher in most cases. It was highest in the case of almost no noise. For this signal a continuous segment of 91% of the signal was found to be silence. The average length is an important parameter because it demonstrates what type of silence is being removed. Savoji's algorithm performs best when used to detect small periods of silence for the purpose of isolating individual words. The proposed algorithm finds the extended consecutive silences that occur while one subscriber is idle during a conversation. This is a critical difference in the context of a video conferencing system.

Figure 9 shows silence with low background noise. Figure 10 shows where Savoji's modified algorithm could not detect and remove silence. The peaks are where speech was detected, while the troughs are where silence was detected. Figure 11 shows the more favorable results for the proposed algorithm. As can be seen, the proposed algorithm is not confused by the difference between background noise and speech. As desired, it finds most of the signal to be silence.

Experiments were also conducted using an actual video conferencing system, Xphone (Eleftheriadis et al. 1994), based on workstations interconnected over an Ethernet network. The simple difference-based algorithm used in Xphone was substituted with the proposed one, with very good results. Most noticeable was the elimination of the frequent false positives in which the algorithm removed segments of speech, mistaking them for silence.

6 Conclusion

We have posed the silence detection problem in a multimedia communications environment. These environments are characterized by their packet-based data handling and communication facilities, as well as a varying degree of QoS support (i.e. bandwidth and delay). We have identified several requirements for efficient use of silence detection, namely elimination of only intersentence (or longer) silence, low complexity, and low decision delay. Due to the particular structure of multimedia communication systems, existing algorithms cannot be easily accommodated.

We have presented a novel algorithm for silence detection, based on the small and large signal behavior of the speech waveform in the μ-law domain. The algorithm exploits the μ-law non-linearity instead of trying

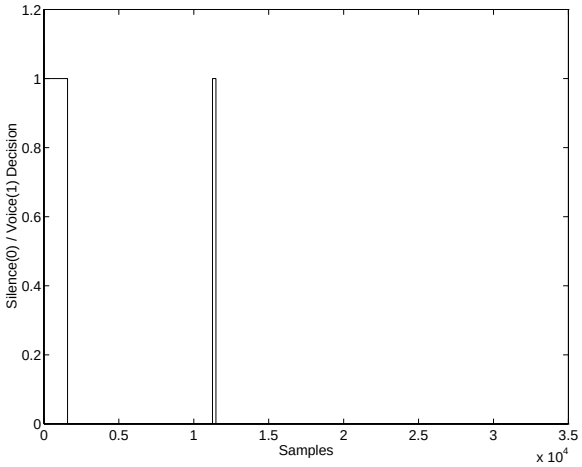
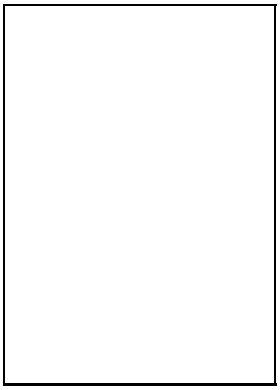


Fig. 11. Proposed algorithm operating on low noise signal

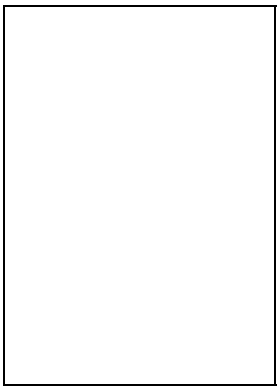
to sidestep it, hence allowing a particularly fast and robust design. It outperforms others in its class of complexity. Aside from the simple magnitude detector, it is the simplest in its class: it has only one filter, performs no μ-law transformations, requires very little memory, and only removes silence through frame truncation. Implementation and experimentation in an actual desktop video conferencing system proved to have diminishing effect on the degradation of the quality of speech, while facilitating a low end-to-end delay.

References

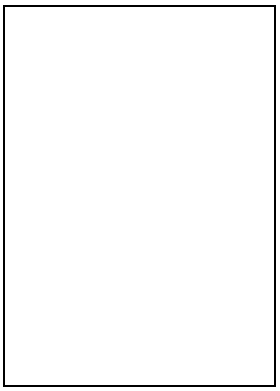
1. Bell Telephone Laboratories (1982) Transmission Systems for Communications, 5th ed.
2. Bullington K, Fraser JM (1959) Engineering Aspects of TASI, BSTJ, 38:353–364.
3. DonVito MB, Schoenherr BW (1985) Subband Coding with Silence Detection, Proceeding of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-85), Tampa, FL, pp. 1433–1436.
4. Drago PG, Molinari AM, Vagliani FC (1978) Digital Dynamic Speech Detectors, IEEE Transactions on Communications, 26:140–145.
5. Eleftheriadis A, Pejhan S, and Anastassiou D (1993) Algorithms and Performance Evaluation of the Xphone Multimedia Communication System, ACM Multimedia Conference, Anaheim, CA, pp. 311–320.
6. Eleftheriadis A, Pejhan S, and Anastassiou D (1994) Architecture and Algorithms of the Xphone Multimedia Communication System, ACM Multimedia Systems Journal, 2:89–100.
7. LeGall D (1991) MPEG: A Video Compression Standard for Multimedia Applications, Communications of the ACM, 34:46–58.
8. Lynch JF, Josenhans LJ, Crochiere RE (1987) Speech/Silence Segmentation for Real-Time Coding via Rule Based Adaptive Endpoint Detection, IEEE International Conference Acoustics, Speech, and Signal Processing (ICASSP-87), Dallas, TX, pp. 1348–1351.
9. Oppenheim AV, Schafer RW, (1989) Discrete Time Signal Processing. Prentice Hall, New Jersey.
10. Rangoussi M, Delopoulos A, Tsatsanis M (1993) On the Use of Higher Order Statistics for Robust Endpoint Detection of Speech, IEEE Signal Processing Workshop on Higher-Order Statistics, South Lake Tahoe, CA, pp. 56–60.
11. Savoji MH (1989) A Robust Algorithm for Accurate Endpointing of Speech Signals, Speech Communication, 8:45–60.
12. Schwartz M, (1987) Telecommunication Networks: Protocols, Modeling, and Analysis, Addison-Wesley, Reading, MA.
13. Un CK and Lee HH (1980) Voiced-Unvoiced-Silence Discrimination of Speech by Delta Modulation, IEEE Transactions on Acoustics, Speech, and Signal Processing, 28:398–407.



Stephen Jacobs received both the B.S. and M.S. in electrical engineering from Columbia University in 1994 and 1995, respectively. He also holds a B.A. in Physics from Bard College.
He is currently a Ph.D. candidate in the Electrical Engineering Department at Columbia University and a Graduate Research Assistant in the Image and Advanced Television Laboratory. His research interests include adaptive multimedia protocols and applications for networks without quality of service guarantees.
In 1996, Mr. Jacobs was awarded the Kodak Fellowship. He is a student member of the IEEE.



Alexandros Eleftheriadis was born in Athens, Greece, in 1967. He received the Diploma in Electrical Engineering and Computer Science from the National Technical University of Athens, Greece, in 1990, and the M.S., M.Phil., and Ph.D. degrees in Electrical Engineering from Columbia University, New York, in 1992, 1994, and 1995 respectively.
Since 1995 he has been an Assistant Professor in the Department of Electrical Engineering at Columbia University, where he is leading a research team working in the areas of visual information representation and compression, video communication systems (including video-on-demand and Internet video), distributed multimedia systems, and the fundamentals of compression. During the summers of 1993 and 1994, he was with AT&T Bell Laboratories, Murray Hill, NJ, developing low bit rate model-assisted video coding techniques for video conferencing applications. From 1990 until 1995, he was a Graduate Research Assistant in the Department of Electrical Engineering at Columbia University.
Dr. Eleftheriadis is a member of the ANSI X3L3.1 Committee, and is participating in the ISO/IEC JTC1/SC29/WG11 (MPEG) standardization activity as well as DAVIC. He is a member of the IEEE, the ACM, and the Technical Chamber of Greece.



Dimitris Anastassiou was born in Athens, Greece, in 1952. He received the Diploma in electrical engineering from the National Technical University of Athens, Greece, in 1974, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1975 and 1979, respectively.
Dr. Anastassiou joined the faculty of Columbia University in 1983, where he is currently Professor of Electrical Engineering, Director of the Image and Advanced Television Laboratory, and Director of the "Columbia New Media Technology Center", a cross-disciplinary research center.
Prior to joining Columbia University in 1983, he was with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member, working on the development of the internal IBM video-conferencing system.
His research interests focus on digital video processing and communications, with emphasis on multimedia applications. He has received an IBM Outstanding Innovation Award and an NSF Presidential Young Investigator Award. He is an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology, and a member of ISO's WG11 Moving Picture Experts Group (MPEG) digital video coding standardization effort. He has been Guest Editor of special issues for various journals and has co-organized MPEG and DAVIC meetings at Columbia University, as well as several workshops and sessions on the subject of digital video and Advanced Television.

This article was processed by the author using the LaTeX style file *cljour2* from Springer-Verlag.