A Bottom-Up Algorithm for Estimating Time-Varying Delays in Coded Speech

Stephen D. Voran

Institute for Telecommunication Sciences
325 Broadway
Boulder, Colorado 80305, USA
svoran@its.bldrdoc.gov 303-497-3839 Voice 303-497-5969 Fax

ABSTRACT

In packetized speech transmission, end-to-end delay can vary, even over short timescales. Estimating the resulting speech delay histories is critical to diagnostic and quality estimation efforts. We present a new bottom-up algorithm for estimating time-varying speech delays. The bottom-up approach is well-suited to real-time implementation. The algorithm works with very low-rate codecs as well as the higher-rate codecs that are more common in VoIP applications. We describe the new algorithm in some detail and provide descriptions of the databases and techniques used to develop and test the new algorithm.

Keywords: speech delay estimation, speech quality estimation, temporal discontinuity, VoIP

1. INTRODUCTION

The packetized transmission of telephone bandwidth speech is gaining prominence in the telecommunications industry. A significant driver of this trend is Voice over Internet Protocol (VoIP) services. In circuit-switched speech transmission, end-to-end transmission delay is nearly always constant, but in packetized speech transmission, end-to-end delay can vary, even over short timescales. The mechanisms that cause this delay variation are well-documented. See [1] and [2] for examples.

There are two main motivations for estimating the delay history of a packetized speech transmission. First, an estimated delay history is required before one can make meaningful input-output based objective estimates of perceived speech quality. (Output-only based estimates of speech quality can enjoy great immunity to delay issues.) Second, knowing the delay history can help to guide the design and optimization of the entire speech transmission system, including the jitter buffer playout algorithm at the receiver.

The delay estimation problem can be described as follows. Given a pair of vectors of speech samples $\boldsymbol{x}$ (system input) and $\boldsymbol{y}$ (system output), for each sample in $\boldsymbol{y}$ find the offset (in samples) to the corresponding sample in $\boldsymbol{x}$. In most practical applications the result is a partition of $\boldsymbol{y}$ into multiple intervals of samples with a single offset value for each interval. The sample rate associated with $\boldsymbol{x}$ and $\boldsymbol{y}$ can be used to convert the offsets in samples to relative delays. The timing relationship between the recording of $\boldsymbol{x}$ and the recording of $\boldsymbol{y}$ can be used to convert these relative delays to absolute delays.

One solution is described in [3] and was first mentioned in [4]. This can be described as a top-down solution since it starts with a single estimate for all of the samples in $\boldsymbol{y}$ and then uses a set of rules to recursively subdivide $\boldsymbol{y}$ into smaller and smaller segments, with the goal of terminating when each of the final segments has a single constant delay. In this paper we propose a bottom-up solution. Here a fixed length delay-estimation window is swept across $\boldsymbol{y}$, resulting in a series of delay estimates at regular intervals across $\boldsymbol{y}$. These estimates are then processed by a median filter which again uses a fixed window that is swept across $\boldsymbol{y}$. When a real-time implementation is desired, data-flow issues make a bottom-up solution more practical than a top-down solution.

We are aware of emerging systems that employ packetized transmission of very low-rate encoded speech. One scenario involves Internet-based interconnection of land-mobile radio systems that use very low-rate speech codecs. This can result in received speech that has both significant codec distortions and non-constant delay. Reference [5] indicates that the solution given in [4] (at least as realized in conjunction with the quality estimation algorithm given in [5]) has not been verified as applicable to this scenario. The algorithm described in this paper was developed to provide reliable delay estimates for very low-rate speech codecs, as well as the more typical VoIP scenario.

The algorithm described here follows from the delay estimation technique that is described in [6] and [7]. In particular, it first uses low resolution techniques to search wide ranges of possible delays, then uses high resolution techniques to search narrower ranges of possible delays, and finally uses a set of rules to combine these results only when advantageous. Matching low resolution with wide searches and high resolution with narrow searches is an inherently efficient approach. The three stage approach also allows for the fact that some systems have a well-defined delay down to the speech sample level, while others simply do not.

In the next section we describe the speech database that was used to develop the algorithm. Then we address the issue of how to realistically measure the performance of a delay estimation algorithm. Next we provide a description

of the new delay estimation algorithm, followed by assessments of its performance.

2. DEVELOPMENT DATABASE

The development and testing of a delay estimation algorithm requires a database of input and output speech files that cover the range of speech, codecs, and network conditions of interest. This database must be accompanied by a corresponding database that holds the true delay history for each of the output speech files. One could build a speech file database by recording real VoIP traffic. This would ensure that the data would be realistic, but there is no way to build a corresponding set of true delay histories. Even if the network were probed and packet delays were measured, there would be no way to account for the operation of the jitter buffer playout algorithm. In addition, one might question whether or not the recordings contain traffic and network conditions that are "typical."

We chose to generate a database using software and hardware tools that allow us to have complete control and knowledge of the true delay history. We start with 72 speech files; 12 files from each of 3 female and 3 male talkers. Each file contains two sentences from the Harvard Phonetically Balanced Sentence Lists [8], is about 7 seconds in length, and is normalized to 26 dB below overload. No intermediate reference system (IRS) [9] filtering is used.

We then pass the 72 files through 7 different speech codecs: 64 kbps G.711 PCM [10], 8 kbps G.729 CS-ACELP [11], 5.3 kbps G.723.1 ACELP [12], 4.6 kbps TETRA ACELP [13], 4.4 kbps IMBE [14], and 2.4 and 1.2 kbps MELP [15]. The IMBE codec is implemented in hardware and uses analog speech input and output. The other six codecs are implemented in software.

The final step is the addition of simulated network impairments to the codec output speech files. One might attempt to use models of network behavior to drive this step. This would require some assumptions to arrive at a relevant set of model parameters. Further, the mapping from network impairments to impairments in coded speech is a function of multiple system parameters including codec type, packet size, packet loss concealment technique, and jitter buffer playout algorithm. The calculation of a true delay history through all these factors would be very complicated. Rather than make a large number of approximations and simplifying assumptions to model networks and then map network impairment parameters to speech impairments, we simply apply impairments directly to the speech signal.

The impairments are applied directly to the output speech signal at four different levels: zero, two, four, or eight impairments per speech file. The intent is to simulate a range of network conditions from perfect to very bad.

The locations of the impairments are randomly chosen for each output file. Thus they may appear in segments of active speech or in silences. The duration of each impairment is randomly chosen to be either 10, 20, or 40 ms, consistent with the values one might typically expect to observe in a packetized speech transmission system. Finally, the type of impairment is randomly chosen from three options: loss, pause, and jump.

For the loss impairment, T =10, 20, or 40 ms of speech is muted, and the Reverse Ordered Replicated Pitch Period (RORPP) packet loss concealment (PLC) algorithm given in [16] is used to form an approximate replacement for the deleted speech. This simulates the situation where one or more packets are lost, concealment is applied, but there is no interruption of the time axis, and no change in the end-to-end delay.

In the pause impairment, T =10, 20, or 40 ms of silence is inserted (just as if a tape player pause button were depressed for only T ms), and the RORPP PLC algorithm is used to extrapolate the speech before the silence in order to cover the silence. This simulates jitter buffer underflow with PLC applied to hide the underflow. The time axis is dilated by T ms, and the end-to-end delay increases by T ms.

The jump impairment requires that T =10, 20, or 40 ms of speech be deleted and the time axis is contracted accordingly. Some smoothing is applied at the discontinuity to minimize audible clicks. This simulates jitter buffer overflow. Since the time axis is contracted by T ms, the end-to-end delay decreases by T ms.

The structure of the development database is summarized in the first 3 columns of Table 1. The database contains a total of 28 conditions (7 codecs × 4 impairment levels/codec), 2016 output speech files (28 conditions × 72 output speech files/condition) and about 4 hours (2016 output speech files × 7 seconds/output speech file) of output speech.

3. COST FUNCTIONS

To develop and test a delay estimation algorithm, one must employ some measure of merit. A measure based solely on the delay estimation error (estimated delay – true delay) may seem to be the obvious initial choice. In our application such measures are not sufficient because they ignore the fundamental uncertainty in the true delay. They also ignore the closely related issue of the variable sensitivity of objective speech quality estimation algorithms to delay estimation errors for different codecs. Figures 1 and 2 provide an example of these two issues and a discussion follows.

The solid lines in the Figures 1 and 2 show estimated speech quality vs. input-output shift in samples for speech that has been passed through a G.711 PCM codec and a 2.4 kbps MELP codec (Codecs 1 and 6 respectively). The speech quality estimates are generated by the MNB speech quality estimation algorithm [6]. A shift of 0 samples provides the highest estimated speech quality for codec 1 and speech quality estimates drop fairly dramatically in that neighborhood. (Secondary peaks appear near shifts of ±60 samples. These shifts correspond to an input-output

misalignment of one average pitch period.) For codec 6, the situation is less clear. There is a range of about 40 samples in the neighborhood of shift=286 that provide approximately equivalent quality estimates. Immediately outside this interval, the estimates drop off slowly compared to codec 1.

The dashed lines in these two figures show simple, smooth, symmetric fits to this data. The data in these figures is slightly asymmetric overall, but the neighborhoods of the peaks are very close to symmetric. In light of this, it seems most appropriate to require the fits to be symmetric, and we fit for a good match near the peak (the most important area for this application) and accept some fitting error on the tails (which are much less important to this work). It is advantageous for the fits to decrease monotonically as one moves away from a peak. Thus the fits do not attempt to model the minor secondary peaks in the data.

The centers of the fits provide a definition of the delay of the codec. We invert and center these fits to form the corresponding cost functions as shown in Figure 3. These cost functions tell approximately how much reduction in estimated speech quality we can expect when we use estimated delays that differ from the true delays as defined above. These approximations are slightly lower than the data in some areas and slightly higher than the data in other areas, but they are close enough to provide the information that we need.

For codec 1 the true delay τ is 0 samples and the cost function $c_1(\hat{\tau}-\tau)$ provides the cost (in terms of reduction of estimated speech quality) associated with using the estimated delay $\hat{\tau}$ in place of the true delay τ. Similarly, for codec 6 the true delay τ is 286 samples and the cost function $c_6(\hat{\tau}-\tau)$ provides the cost of using $\hat{\tau}$ in place of τ. Note that for codec 6 when the estimated delay is close to the true delay, the cost is zero. This is consistent with the uncertain nature of the true delay. We can say that codec 1 has little delay uncertainty and high delay sensitivity in the neighborhood of the true delay. Codec 6 has greater delay uncertainty, and lower delay sensitivity in the neighborhood of the true delay.

For each of the 7 codecs in the development database we use the method outlined above to find a true delay and a cost function $c_i(\hat{\tau}-\tau)$, i=1 to 7. Cost can be a weak function of talker and speech material and our technique generates cost functions that have been averaged across these variables. The MNB output is a logistic function applied to an auditory distance (L(AD)). This L(AD) speech quality scale ranges from zero to one. Thus $c_i(\hat{\tau}-\tau)=0.01$ indicates that for codec i, using the delay $\hat{\tau}$ instead of the true delay τ will cause a drop in estimated speech quality that is 1% of the full quality scale. More generally, we can interpret $c_i(\hat{\tau}-\tau)$ as the fraction of the full quality scale that estimated quality will drop when the delay $\hat{\tau}$ is used instead of the true delay τ.

These cost functions are used to assess the merit of estimated delay histories. Assume that the k^{th} file pair has been passed through the i^{th} codec and is then further impaired as described in Section 2. We can calculate the true delay history $\tau_k(j)$ for this file by combining the true codec delay with the known random locations and durations of the impairments. An estimated delay history $\hat{\tau}_k(j)$ results in an average cost Δ_k of

$$\Delta_k = \frac{1}{N}\sum_{j\in A} c_i\left(\hat{\tau}_k(j)-\tau_k(j)\right), \qquad (1)$$

where A is the set of all active speech samples in the system output speech vector $\boldsymbol{y}$ and N is the number of samples in A. We then average this result over all of the M speech files in a given condition to find the per-condition average cost

$$\bar{\Delta}_c = \frac{1}{M}\sum_{k=1}^{M}\Delta_k \ . \qquad (2)$$

4. ALGORITHM FOR ESTIMATING VARIABLE DELAYS

A fundamental trade-off permeates time delay estimation: longer delay estimation windows generally provide more robust delay estimates, but longer windows also make it more difficult to respond quickly to changes in delay. Shorter delay estimation windows generally provide less robust delay estimates, but shorter windows also make it easier to respond quickly to changes in delay. A second trade-off involves implementations of cross-correlations: compared to direct-form cross correlations, FFT-based cross correlations can be more efficient, but they also require longer windows in order to robustly search a given range of delays. An additional trade-off is often seen: the selection of an algorithm parameter value (e.g., processing window size, correlation threshold) is often a compromise between a value that favors the higher rate codecs and a value that favors the lower rate codecs. The specific values we report here are the result of optimization over the 28 conditions in the development database.

Developing an effective, robust, efficient delay estimation algorithm requires successful resolution of these trade-offs along with many others. To insure broad applicability, we monitor the average cost of delay estimation error for each of the 28 conditions throughout the algorithm development process.

Figure 4 provides a summary of the core components of the new algorithm for estimating time-varying delays in coded speech. A single coarse average delay is first calculated for the entire signal vector $\boldsymbol{y}$ using smoothed speech envelopes. A delay tracking stage then uses higher resolution envelopes and seeks to follow variations about the average delay value. Median filtering is then used to improve the accuracy of these results, and a refinement stage refines

each delay estimate where possible. The final output is a vector containing delay estimates spaced at 40 ms intervals across the entire duration of the output speech signal $\boldsymbol{y}$. Each estimate has full resolution, i.e., it is to the nearest sample.

4.1 Level Normalization

The two input vectors to the algorithm described in Figure 4 contain samples of system input speech and system output speech. The sample rate is 8000 samples/second. The level normalization stages normalize these vectors of speech samples to an active speech level of 26 dB below overload using a technique motivated by that given in [17]. The resulting level-normalized system input and output speech vectors are $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively**.**

4.2 Coarse Average Delay Estimation

Next, a coarse estimate of average delay is calculated from speech envelopes. To create the envelopes, $\boldsymbol{x}$ and $\boldsymbol{y}$ are rectified and low pass filtered. The filter is an order 400 FIR low-pass filter with 48 dB of attenuation at 62.5 Hz. This filter largely eliminates pitch information, leaving a very smooth temporal speech amplitude envelope. The resulting signals are then sub-sampled by a factor of 64, resulting in speech envelopes sampled at 125 samples/second. These envelopes are then passed to the coarse average delay estimation stage which applies an FFT-based cross correlation to estimate the average delay τ_0 between the two speech envelopes. The resolution of this delay is ±64 samples in the original (8000 samples/second) speech domain. The corresponding peak correlation value is ρ_0.

4.3 Speech Activity Detection

The speech activity detection stage classifies each sample of $\boldsymbol{y}$ as either active speech or inactive so that samples in each class can be treated appropriately. To make this classification, the speech envelope of $\boldsymbol{y}$ described above is compared with a fixed threshold of 56. Regions above this threshold are classified as active and those regions are then extended 100 ms forward and backward in time to create the final activity classification.

4.4 Delay Tracking

The delay tracking stage estimates the actual delay of the samples in $\boldsymbol{y}$ as that delay varies about the average delay τ_0. This stage uses speech envelopes with 250 Hz bandwidths. To create the envelopes, $\boldsymbol{x}$ and $\boldsymbol{y}$ are rectified and low pass filtered. The filter is an order 128 FIR low-pass filter with 51 dB of attenuation at 250 Hz. This filter preserves much of the pitch information in the speech signal, thus allowing for its potential use in delay estimation. The resulting signals are then sub-sampled by a factor of 16, resulting in speech envelopes sampled at 500 samples/second. These envelopes are then passed to the delay tracking stage. This stage applies a direct-form (non FFT-based) cross correlation algorithm that uses 150 ms segments of the envelope of $\boldsymbol{y}$. This window is swept across the envelope of $\boldsymbol{y}$ in 40 ms steps. This step size was selected as a good compromise between algorithm speed and algorithm accuracy. The cross-correlation is calculated when at least 10% of the samples in the window are classified as active. The corresponding search window in the envelope of $\boldsymbol{x}$ is centered on τ_0 (the delay value produced by the coarse average delay estimation stage) and extends 200 ms to either side of τ_0, allowing for the tracking of delay deviations as great as 200 ms. The result of the delay tracking stage is a vector $\boldsymbol{\tau_1}$ of delay estimates spaced at 40 ms intervals in the active portions of the speech signal. Each estimate has a resolution of ±16 samples in the original domain.

The median filtering stage sweeps a 500 ms window across $\boldsymbol{\tau_1}$. It calculates the median value of all valid delay estimates in the window that have a correlation of at least 0.8. This thresholding helps to minimize the effect that lower quality delay estimates can have on the filter output. This window is advanced in 40 ms steps. The median filter tends to reject small groups of markedly different delay estimates which are often erroneous. It also tends to generate a more piecewise-constant delay estimation history and this is consistent with the piecewise-constant nature of the true delay histories in packet speech transmission systems. The result of the median filtering stage is a vector $\boldsymbol{\tau_2}$ containing delay estimates spaced at 40 ms intervals. These estimates are available only in active areas of speech where at least some of the correlations exceed 0.8. Each estimate has a resolution of ±16 samples in the original domain.

4.5 Refinement of Delay Estimates

The delay refinement stage seeks to refine the delay estimates available in $\boldsymbol{\tau_2}$ to the sample level where possible. This stage uses the absolute values of the samples in $\boldsymbol{x}$ and $\boldsymbol{y}$ and applies a correlation operation to each segment of constant delay in $\boldsymbol{\tau_2}$ that contains at least 10 ms of active speech. The use of the absolute value function makes the algorithm completely robust to 180 degree phase changes in the speech signals. The correlation operation searches a range of ±72 samples about the delay given in $\boldsymbol{\tau_2}$. When a segment is at least 200 ms in length, an FFT-based cross correlation is used. Otherwise a direct-form correlation is used. If the segment is one second in length or greater, the refinement is very likely to be reliable and it is used without further testing. For shorter length segments, only those with a correlation of 0.7 or greater are retained. The result of the delay refinement stage is a vector $\boldsymbol{\tau_3}$ containing delay estimates spaced at 40 ms intervals. Valid estimates are available consistent with the rules described above, and each estimate has full resolution, i.e., it is to the nearest sample in the original (8000 samples/second) domain.

4.6 Short Segment Correction

The next stage seeks to correct shorter segments of constant delay that may be erroneous. The rules of this stage are driven by the knowledge that the true delay history is piecewise constant. Three different types of short segments are treated and examples of each are shown in Figure 5.

A "pulse" is a segment with estimated delay τ_p that has two valid immediate neighbors (left and right) that share a common delay estimate $\tau_n \neq \tau_p$. Figure 5 shows a 200 ms pulse where τ_p=30 ms and τ_n=40 ms. When a pulse is 280 ms or shorter it is likely that the estimated delay change from τ_n to τ_p and back to τ_n again is erroneous, so the pulse and its two neighbors are replaced with a single segment with a constant estimated delay of τ_n.

A "step" is a segment with estimated delay τ_s that has two valid immediate neighbors (left and right) with different delay estimates $\tau_l \neq \tau_r$, both of which differ from τ_s. Figure 5 shows a step with 80 ms duration where τ_l=40 ms, τ_s=70 ms, and τ_r=80 ms. When a step is 80 ms or shorter it is likely that the true delay actually changes directly from τ_l to τ_r without passing through τ_s. To test this, the rectified speech in $\boldsymbol{y}$ corresponding to the step segment is correlated with the corresponding rectified speech in $\boldsymbol{x}$ using delays τ_l ,τ_r , and τ_s. If the delay τ_l produces the highest correlation, then the step segment is combined with its left neighbor and the entire new segment is assigned a delay estimate of τ_l. If the delay τ_r produces the highest correlation, then the step segment is combined with its right neighbor and the entire new segment is assigned a delay estimate of τ_r. If the delay τ_s produces the highest correlation, then the step segment is left unchanged.

A "tail" is a segment with an estimated delay τ_t that has only one valid immediate neighbor. This neighbor has a delay estimate of $\tau_n \neq \tau_t$. This situation most often happens at the start or end of an active speech interval. Figure 5 shows a tail with duration 160 ms at the end of an active speech interval where τ_n =80 ms and τ_t =75 ms. When a tail segment is 160 ms or shorter it is likely to be erroneous so it is combined with its valid neighbor and the entire new segment is assigned a delay estimate of τ_n.

The result of the short segment correction stage is a vector $\boldsymbol{\tau_4}$ containing delay estimates spaced at 40 ms intervals. Valid estimates are available consistent with the rules described above and each estimate has full resolution in the original domain.

4.7 Extension of Valid Estimates

The vector $\boldsymbol{\tau_4}$ contains delay estimates that cover almost all active portions of the speech signal, with the possible exception of locations where severe localized impairments have corrupted the speech signal in $\boldsymbol{y}$ to the extent that delay estimation is simply not possible. As a practical matter, it is sometimes desirable to have a delay estimate for every single sample in the speech vector $\boldsymbol{y}$, including samples in silent intervals where there is no speech signal upon which to base an estimate. The final stage of the algorithm extends valid delay estimates to cover segments where no valid estimate has yet been calculated. When such a segment has two neighbors (left and right) with valid delay estimates, then that segment is divided into a left half and a right half. The valid delay estimate for the left neighboring segment is extended to cover the left half. The valid delay estimate for the right neighboring segment is extended to cover the right half. This stage generates the final output of the variable delay estimation algorithm. This output is a vector $\boldsymbol{\tau_v}$ containing delay estimates spaced at 40 ms intervals across the entire duration of the output speech signal $\boldsymbol{y}$. Each estimate has full resolution, i.e., it is to the nearest sample.

4.8 Real-Time Implementation

In a real-time implementation $\boldsymbol{x}$ and $\boldsymbol{y}$ would contain the most recent block of samples from continuing system input and system output speech streams. Once the current delay has been established, the coarse average delay estimation stage is not needed; the delay tracking stage can center its delay search window using previous outputs of the algorithm. The other core components of the algorithm would remain largely unchanged.

5. ALGORITHM FOR ESTIMATING FIXED DELAYS

Since longer delay estimation windows provide more robust delay estimates, it is advantageous to use the longest appropriate window. When it is known a priori that there is a single fixed time delay between the speech samples in $\boldsymbol{x}$ and $\boldsymbol{y}$, we use a specialized algorithm. This algorithm performs the level normalization described in 4.1 above, and then the coarse average delay estimation described in 4.2 above. This stage results in a delay estimate with a resolution of ±64 samples.

The next stage performs an FFT-based cross correlation on rectified versions of $\boldsymbol{x}$ and $\boldsymbol{y}$ to create the correlation waveform $\rho(\tau)$. In some cases the peak of this waveform provides useful information, but in other cases, $\rho(\tau)$ must be smoothed before a useful peak can be extracted. In the development of this stage we consider $\rho(\tau)$ and 3 smoothed versions of $\rho(\tau)$ with nominal bandwidths of 125, 62.5, and 31.25 Hz. We consider how the peak value of $\rho(\tau)$ might be used to select between these four versions in an optimal way.

In the following, only delays that are within 128 samples of τ_0 (the coarse average delay estimate) are considered. If the peak value in $\rho(\tau)$ is 0.73 or greater, that peak location provides the final delay estimate. Otherwise, the system that produced $\boldsymbol{y}$ is not one that preserves waveforms and a more robust peak finding algorithm is desirable. If the peak correlation value in $\rho(\tau)$ is in the interval [0.67, 0.73) then $\rho(\tau)$ is smoothed (using an order 192 FIR low-pass filter with 6 dB of attenuation at 62.5 Hz and 40 dB of attenuation at 125 Hz) and the location of the resulting peak provides the final delay estimate. If the peak correlation value in $\rho(\tau)$ is less than 0.67, then $\rho(\tau)$ is smoothed more dramatically (using an order 384 FIR low-pass filter with 6 dB of

attenuation at 31.25 Hz and 40 dB of attenuation at 62.5 Hz) and the location of the resulting peak provides the final delay estimate. This stage results in a single scalar delay estimate τ_f. The resolution of τ_f depends on the rules described above, and it can be as high as full resolution.

6. COMPLETE ALGORITHM

If a priori information regarding the variable or fixed nature of the delay between $\boldsymbol{x}$ and $\boldsymbol{y}$ is available, one can select and apply the variable delay estimation of section 4 or the fixed delay estimation algorithm of section 5 appropriately. In many situations such information is available, but in others it is not. Several options are available when it is not known if the delay is fixed or variable.

Since fixed delay is a special case of variable delay, one can simply apply the variable delay estimation algorithm. But since the variable delay estimation algorithm does not maximize the time delay estimation window, the results may not always be as robust as desired, especially for lower rate codecs. The development database includes 7 fixed delay conditions. Conditions 1, 5, 9, and 13 are treated equally well by the variable and the fixed delay estimation algorithms. Conditions 17, 21, and 25, however, show about 0.004 higher average cost when treated with the variable delay estimation algorithm rather than the fixed delay estimation algorithm. Note that these three conditions are lower rate codecs that do not tend to preserve waveforms as well as the higher rate codecs.

A second option would be to apply both algorithms and then apply additional tests to determine which is most reasonable. One could apply an objective estimator of speech quality and select the delay estimation result that gives the highest estimated quality.

We simplify and augment this second option to integrate the two algorithms into a single complete algorithm. The complete algorithm is summarized in Figure 6. When no prior information is available, the complete algorithm switches appropriately between the fixed and variable delay estimation algorithms. This switching is driven by speech envelope correlations and log spectral error (LSE) measurements.

When the average coarse delay estimation stage produces a correlation value ρ_0 that is less than 0.96, there is a very high probability that the true delay is variable and the complete algorithm pursues the variable delay path only. When this correlation value is 0.96 or greater, the true delay may be fixed or variable, and both the fixed and variable delay estimation paths are pursued.

The resulting fixed and variable delay estimates are used to create two delay-compensated versions of $\boldsymbol{x}$ called $\boldsymbol{x_f}$ and $\boldsymbol{x_v}$ respectively. If the test $\mathrm{LSE}(\boldsymbol{x_v}, \boldsymbol{y}) < \mathrm{LSE}(\boldsymbol{x_f}, \boldsymbol{y})$ is satisfied, then the variable delay estimate is selected, otherwise the fixed delay estimate is selected. Here LSE denotes a conventional log spectral error that uses a 16 ms periodic Hanning window. LSE windows are adjacent, but are not overlapped.

7. ALGORITHM PERFORMANCE

The performance of the new delay estimation algorithm on the development data is summarized in Table 1 using the per-condition average cost values defined in (2). Since the algorithm does not perfectly estimate every change in delay, the average cost associated with the delay estimates increases with the number of impairments. In other words, as we add additional changes in delay and additional packet losses, it gets harder to estimate the delay properly. The highest average cost found on the development data is 0.0265, corresponding to an average drop of 2.65% of the full quality scale. The average cost of error across all 28 conditions in the development database is 0.0101, corresponding to an average drop of 1.01% of the full quality scale. If a fixed estimate of delay is used for each speech file, the average cost of error across all 28 conditions in the development database is 0.1074, and this is about 10 times higher than what is achieved with the new algorithm.

The development database is used throughout the development and optimization of the algorithm so one might expect that it represents a best-case scenario. To test the generality of this algorithm we employ a testing database. This database starts with 128 speech files, 16 files from each of 4 female and 4 male talkers. None of these talkers was used in the development database. For each talker, 8 files are filtered with the IRS transmit filter [9] and the other 8 are filtered with a bandpass filter with approximately flat passband extending from 200 to 3400 Hz. The use of both IRS and flat filtered speech gives the testing database additional breadth of applicability. Each file contains two sentences from the Harvard Phonetically Balanced Sentence Lists [8], is about 7.2 seconds in length, and is normalized to 26 dB below overload.

We then pass the 128 files through twelve codecs that are summarized in column 1 of Table 2. These include the seven codecs used in the development database and five additional codecs. The five additional codecs are 64 kbps G.711 PCM [10], 32 kbps G.726 ADPCM [18], 16 kbps G.728 LD-CELP [19], 8 kbps IS-54 VSELP [20], and 2.45 kbps AMBE which is a "half-rate version" of [14]. All five of these are hardware implementations, and all but the AMBE codec use analog speech input and output. Analog interfaces can introduce additional impairments beyond those of the digital speech codec, thus adding additional challenges to the delay estimation problem. By including four codecs with analog interfaces, we create a more challenging and more realistic testing database.

The testing database uses a new set of random impairment locations. The impairment types are again loss, pause and jump, but are selected according to a new random sequence. The impairment durations are again 10, 20, or 40

ms, but are selected according to a new random sequence. (We consider these values to be most relevant in light of typical VoIP system design. However, they are also the values used in the development database. Thus, in order to most fully test the robustness of the new algorithm, we create a second version of the testing database. In this second version, impairment durations are randomly drawn from the uniform distribution on the interval [1,400] samples, which corresponds to approximately 0 to 50 ms. When 8 impairments are used in a speech file, 50 ms impairment durations can lead to a total delay range of up to 8 × 50 = 400 ms. This is a good match to the delay tracking stage which searches 200 ms on either side of τ_0, for a total search range of 400 ms.)

The testing database contains over 12 hours of speech in 6144 output speech files. In order to evaluate the algorithm performance on the testing database, we first generate per-codec cost functions $c_i(\hat{\tau}-\tau)$, i=1 to 12.

Table 2 contains per-condition average cost values defined in (2) for the testing database and these are shown in Figure 7 as well. As before, the average cost associated with the estimated delay history rises for each codec as the number of impairments is increased. The highest average cost is 0.0214. This worst-case situation is found when 8 impairments are added to each 7.2 second file of 4.4 kbps IMBE (with analog input and output) coded speech. The average cost of error across all 48 conditions in the testing database is 0.0079, corresponding to an average drop of only 0.8% of the full MNB L(AD) speech quality estimation scale. (The results for the second testing database with impairment durations uniformly distributed on (0,50] ms are very similar. The average cost of error across all 48 conditions in the second testing database is 0.0081.)

The results for the three databases are rather similar, indicating that the new delay estimation algorithm is not overly specific to the development database, and that the algorithm is applicable to a very wide range of telephone-bandwidth speech codecs. We consider these results to be satisfactory. The average (0.8%) and even the worst-case (2.1%) perturbations in estimated speech quality for the testing data seem acceptable in light of the fact that the most carefully controlled subjective speech quality tests may attain 95% confidence intervals on the order of ±0.1 units of a 4 unit scale. In other words, experimental results may differ from the underlying true quality value by ±2.5% of the full quality scale. When one considers the additional error inherent in objectively estimating speech quality, it seems likely that the errors attributable to this delay estimation algorithm will make up only a very small portion of the typical total error in objective speech quality estimates.

Comparison of results for codecs 1 (G.711 PCM software implementation) and 8 (G.711 hardware implementation with analog speech input and output) demonstrates that the addition of analog interfaces does make the variable delay estimation problem more challenging. For the three conditions with varying delay, codec 8 shows a modest increase in average cost relative to codec 1.

Figure 8 provides a worse-than-average example of a true delay history and the corresponding estimated delay history for condition 48 of the testing database. Condition 48 is a challenging one; it uses the 2.45 kbps AMBE codec with 8 added impairments per speech file. One sentence is shown, and four delay-change impairments are visible in the true delay history. The estimated delay history generally tracks the true delay history but deviations are clearly visible. The average cost associated with these deviations is Δ_k =0.0351, which is more than twice the average for this condition (0.0154) given in Table 2. Thus our description of this example as a "worse-than-average" example.

The present implementation of the new delay estimation algorithm is via unoptimized Matlab® code. On average, this implementation of the complete algorithm shown in Figure 6 requires a processing time that is only about 0.18 × real time on a PC using a 3.2 GHz Pentium® 4 processor. That is, on average it takes about 1.26 seconds for this implementation to process a pair of 7 second speech files. If it is known a priori that the delay is fixed, the algorithm requires about 0.07 × real time. If it is known a priori that the delay is variable, the algorithm requires about 0.16 × real time. The LSE calculation runs in about 0.01 × real time.

8. CONCLUSION

We have described the development and testing of a bottom-up algorithm for estimating time-varying delays in coded speech. Consistent with the goal, the algorithm is applicable to lower-rate codecs that are used in land mobile radio communication systems as well as to typical VoIP codecs. This breadth of applicability is an improvement over the current state of the art. Because the algorithm takes a bottom-up approach, it is suitable for real time implementation and we consider this to be an additional contribution.

Measuring delay estimation error in samples holds little relevance when such a wide range of codecs is considered. Thus our development and testing is carried out in the context of cost functions. These functions relate errors in delay estimation to reductions in objective estimates of perceived speech quality. To maximize relevancy to actual testing environments, our testing includes 12 codecs with bit rates ranging from 1.2 to 64 kbps and 5 of these codecs use analog speech inputs and outputs. In addition, we employ a wide range of simulated network conditions and a standardized packet loss concealment algorithm.

When there are no variations in delay over the course of a 7 second speech file (simulating good network conditions), we find that delay estimation error from the new algorithm will reduce objective speech quality estimates by as little as zero and as much as 0.5% of the full quality scale, depending on the speech codec. When 8 impairments are added to each file (simulating very bad network conditions),

this range becomes 1.1% to 2.0%. The average across the entire testing database is a drop of 0.8% of the full quality scale. We argue that reductions like these are not likely to be significant in light of the other sources of error in the objective estimation of perceived speech quality.

REFERENCES

[1] M. Hassan and A. Nayandoro, "Internet telephony: services, technical challenges, and products," IEEE Communications Magazine, vol. 38, no. 4, pp. 96-103, Apr. 2000.

[2] S. Voran, "Perception of Temporal Discontinuity Impairments in Coded Speech - A Proposal for Objective Estimators and Some Subjective Test Results," Proceedings of the 2nd International Conference on Measurement of Speech and Audio Quality in Networks, Prague, Czech Republic, May 2003.

[3] J.G. Beerends, A.W. Rix, M.P. Hollier, and A.P Hekstra, "Perceptual evaluation of speech quality (PESQ) The new ITU standard for end-to-end speech quality assessment, Part I – Time-Delay Compensation," J. Audio Eng. Soc., vol. 50, no. 10, pp. 755-764, Oct. 2002.

[4] A.W. Rix and M.P. Hollier, "The perceptual analysis measurement system for robust end-to-end speech quality assessment," Proc. 2000 IEEE International Conference on Acoustics, Speech and Signal Processing, Istanbul, June 2000.

[5] ITU-T Recommendation P.862, "Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs," Geneva, 2001.

[6] S. Voran, "Objective Estimation of Perceived Speech Quality, Part I: Development of the Measuring Normalizing Block Technique," IEEE Transactions on Speech and Audio Processing, July 1999.

[7] ITU-T Recommendation P.931, "Multimedia communications delay, synchronization and frame rate measurement," Geneva, 1998.

[8] IEEE Recommended practice for speech quality measurements, *IEEE Trans. Audio and Electroacoustics*, vol. AU-17, no. 3, pp. 225-246, Sep. 1969.

[9] ITU-T Recommendation P.48, "Specification for an Intermediate Reference System," Geneva, 1988

[10] ITU-T Recommendation G.711, "Pulse code modulation (PCM) of voice frequencies," Geneva, 1988.

[11] ITU-T Recommendation G.729, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)," Geneva, 1996.

[12] ITU-T Recommendation G.723.1, "Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s," Geneva, 1996.

[13] ETSI European Telecommunication Standard 300 395-2, "Radio Equipment and Systems (RES); Trans-European Trunked Radio (TETRA); Speech codec for full-rate traffic channel; Part 2: TETRA codec," Sophia Antipolis – Valbonne, France, 1996.

[14] ANSI/TIA-102.BABA, "APCO Project 25 Vocoder Description," 1998.

[15] T. Wang, K. Koishida, V. Cuperman, A. Gersho, J. Collura, "A 1200/2400 bps coding suite based on MELP," Proceedings of the 2002 IEEE Speech Coding Workshop, Ibaraki, Japan, October, 2002.

[16] ITU-T Recommendation G.711 Appendix I, "A high quality low-complexity algorithm for packet loss concealment with G.711," Geneva, 1999.

[17] ITU-T Recommendation P.56, "Objective measurement of active speech level," Geneva, 1993.

[18] ITU-T Recommendation G.726, "40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)," Geneva, 1990.

[19] ITU-T Recommendation G.728, "Coding of speech at 16 kbit/s using low-delay code excited linear prediction," Geneva, 1992.

[20] I.A. Gerson and M.A. Jasiuk, "Vector sum excited linear prediction (VSELP) speech coding at 8 kbps," Proc. 1990 IEEE International Conference on Acoustics, Speech and Signal Processing, Albuquerque, April 1990.

Note:

Certain commercial equipment and materials are identified in this paper to specify adequately the technical aspects of the reported results. In no case does such identification imply recommendations or endorsement by the National Telecommunications and Information Administration, nor does it imply that the material or equipment identified is the best available for this purpose.

Table 1. Conditions in development database and average cost of the delay estimation error for each condition.

Condition Number	Codec Number and Description	Impairments per File	Average Cost, $\bar{\Delta}_c$ (fraction of quality scale)
1	1 - G.711 PCM, 64 kbps	0	0.0000
2	"	2	0.0034
3	"	4	0.0068
4	"	8	0.0137
5	2 - G.729 CS-ACELP, 8 kbps	0	0.0019
6	"	2	0.0072
7	"	4	0.0117
8	"	8	0.0173
9	3 - G.723.1 ACELP, 5.3 kbps	0	0.0001
10	"	2	0.0020
11	"	4	0.0042
12	"	8	0.0092
13	4 - TETRA ACELP, 4.6 kbps	0	0.0001
14	"	2	0.0027
15	"	4	0.0063
16	"	8	0.0109
17	5 - IMBE, 4.4 kbps, w/ analog I/0	0	0.0043
18	"	2	0.0119
19	"	4	0.0144
20	"	8	0.0196
21	6 - MELP, 2.4 kbps	0	0.0097
22	"	2	0.0170
23	"	4	0.0213
24	"	8	0.0265
25	7 - MELP, 1.2 kbps	0	0.0064
26	"	2	0.0134
27	"	4	0.0179
28	"	8	0.0218

Table 2. Conditions in testing database and average cost of the delay estimation error for each condition.

Condition Number	Codec Number and Description	Impairments per File	Average Cost, $\bar{\Delta}_c$ (fraction of full quality scale)
1	1 - G.711 PCM, 64 kbps	0	0.0000
2	"	2	0.0036
3	"	4	0.0078
4	"	8	0.0159
5	2 - G.729 CS-ACELP, 8 kbps	0	0.0011
6	"	2	0.0040
7	"	4	0.0088
8	"	8	0.0136
9	3 - G.723.1 ACELP, 5.3 kbps	0	0.0000
10	"	2	0.0024
11	"	4	0.0050
12	"	8	0.0119

Table 2 continued.

Condition Number	Codec Number and Description	Impairments per File	Average Cost, $\bar{\Delta}_c$ (fraction of full quality scale)
13	4 - TETRA ACELP, 4.6 kbps	0	0.0000
14	"	2	0.0034
15	"	4	0.0054
16	"	8	0.0114
17	5 - IMBE, 4.4 kbps, w/ analog I/0	0	0.0053
18	"	2	0.0105
19	"	4	0.0143
20	"	8	0.0214
21	6 - MELP, 2.4 kbps	0	0.0028
22	"	2	0.0091
23	"	4	0.0111
24	"	8	0.0167
25	7 - MELP, 1.2 kbps	0	0.0030
26	"	2	0.0088
27	"	4	0.0125
28	"	8	0.0178
29	8 - G.711 PCM, 64 kbps, w/ analog I/0	0	0.0000
30	"	2	0.0042
31	"	4	0.0083
32	"	8	0.0192
33	9 - G.726 ADPCM, 32 kbps, w/ analog I/0	0	0.0000
34	"	2	0.0038
35	"	4	0.0081
36	"	8	0.0187
37	10 - G.728 LD-CELP, 16 kbps, w/ analog I/0	0	0.0000
38	"	2	0.0036
39	"	4	0.0070
40	"	8	0.0152
41	11 - VSELP, 8 kbps, w/ analog I/0	0	0.0002
42	"	2	0.0037
43	"	4	0.0067
44	"	8	0.0138
45	12 - AMBE, 2.45 kbps	0	0.0027
46	"	2	0.0079
47	"	4	0.0111
48	"	8	0.0154

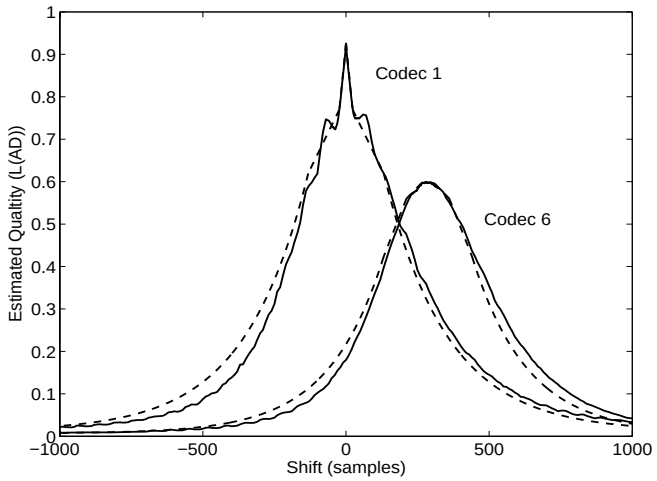

Figure 1. Estimated speech quality versus input-output speech signal shift for two codecs. Raw data (solid line) and simple, smooth, symmetric fit (dashed line).

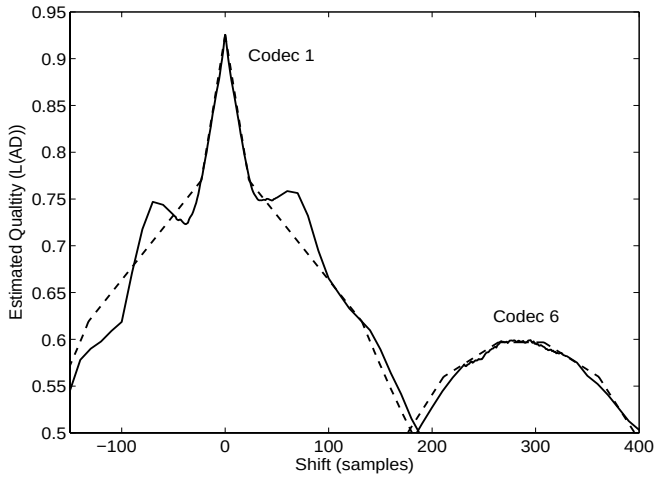

Figure 2. Detail of Figure 1.

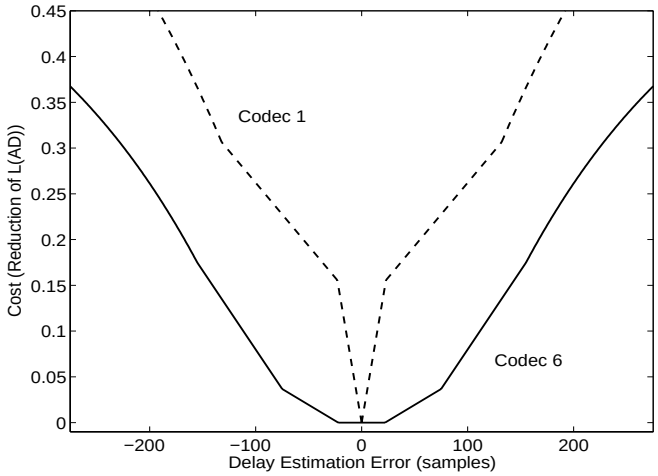

Figure 3. Detail of cost functions for two codecs, derived by inverting and centering the fitted curves in Figure 2.

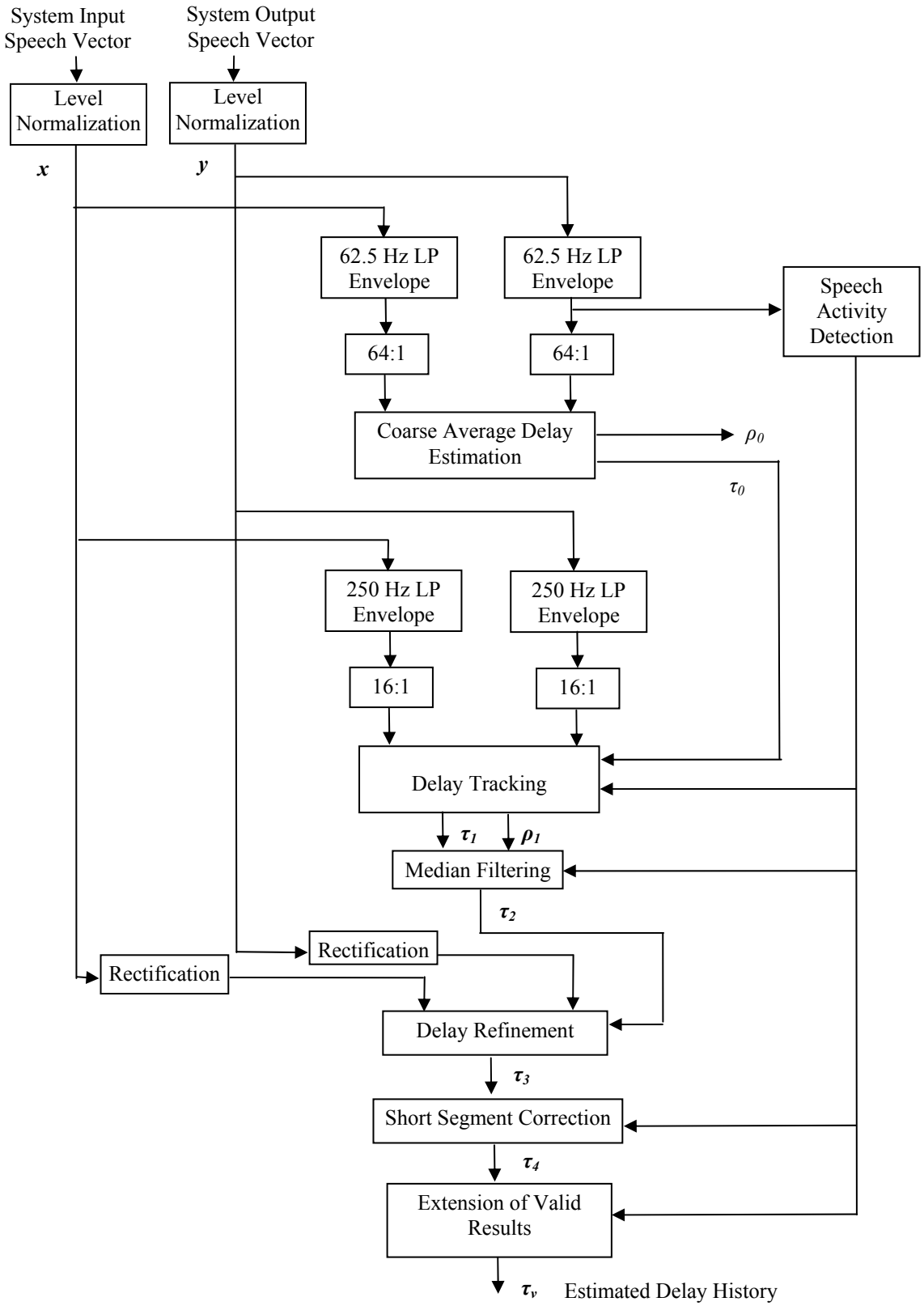

Figure 4. Conceptual block diagram for a bottom-up estimator of time varying delays in coded speech.

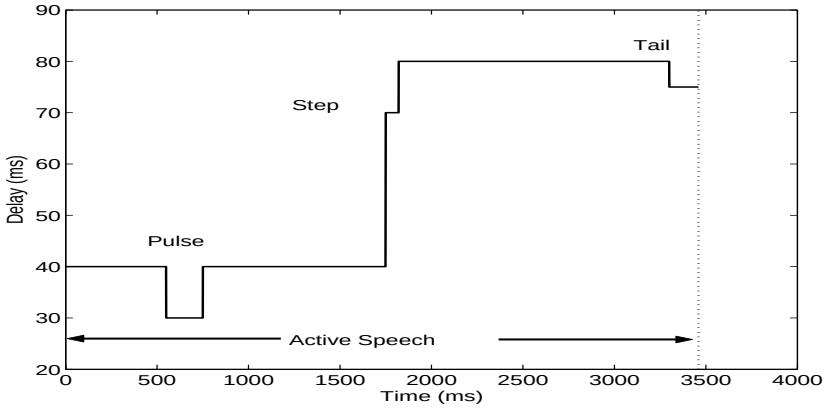

Figure 5. Example of a pulse, a step, and a tail in an estimated delay history.

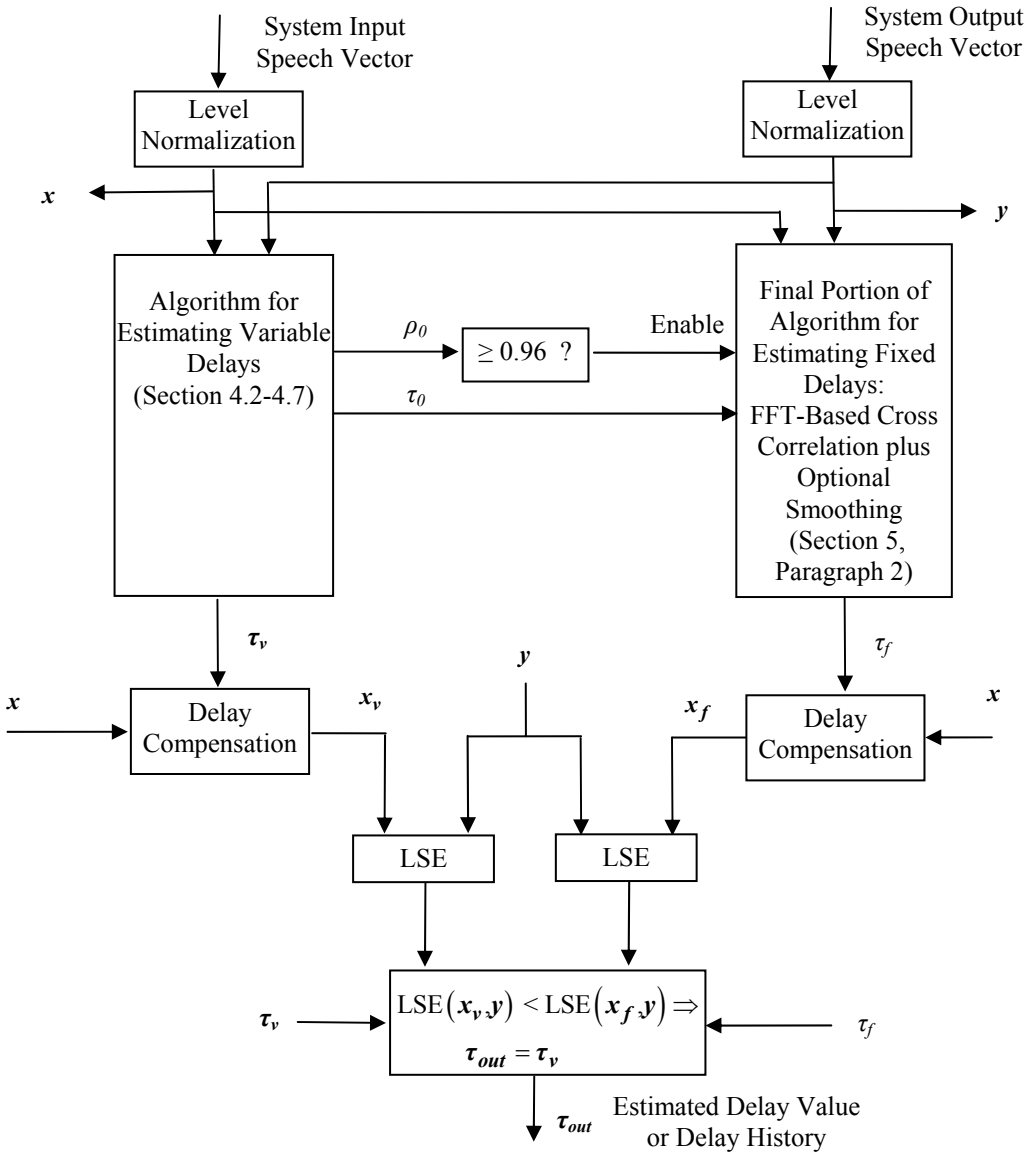

Figure 6. Conceptual block diagram for complete algorithm for estimating fixed or time varying delays in coded speech.

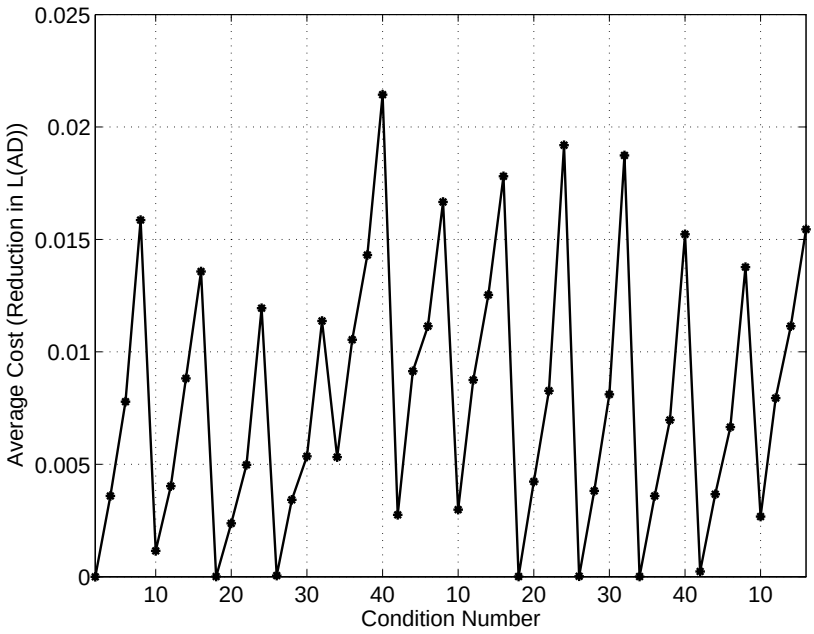

Figure 7. Per-condition average cost of error for the 48 conditions in the testing database.

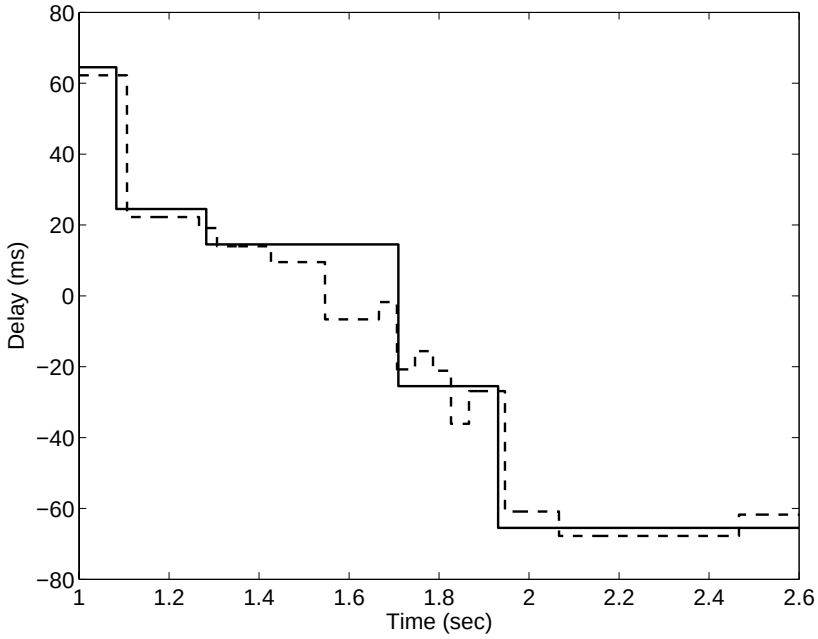

Figure 8. Example true (solid line) and estimated (dashed-line) delay histories for one sentence passed through 2.45 kbps AMBE codec with 8 added impairments per file.