ITU-T SG 12 - STUDY PERIOD 1993-1996 DOC. SQ - 75 .96
EXPERTS GROUP ON SPEECH QUALITY

TITLE: An Algorithm for Estimating the Delay of Telephony Speech

SOURCE: Institute for Telecommunication Sciences
National Telecommunications and Information Administration
U.S. Department of Commerce

DATE: September 26, 1996

AUTHOR: Stephen Voran

CONTACT: Stephen Voran
NTIA/ITS.N3
325 Broadway
Boulder, CO 80303 USA
1-303-497-3839 (Phone)
1-303-497-5323 (Fax)
sv@bldrdoc.gov

ABSTRACT: This contribution is provided for informational purposes. It contains a description of an algorithm that has sucessfully been used to estimate the delay of telephony band speech. The algorithm features a coarse stage that uses speech envelopes and a fine stage that uses speech power spectral densities. Observations on the performance of the algorithm and its potential for extension beyond telephone band speech are offered. This algorithm is included in the Draft ANSI T1 Standard on Multimedia Communications Delay, Synchronization, and Frame Rate Measurement.

An Algorithm for Estimating the Delay of Telephony Speech

1. Introduction

This informational contribution contains a description of an algorithm that the Institute for Telecommunication Sciences (ITS) has developed for estimating the delay of telephony band speech. Section 2 of this contribution provides background information, motivating factors, and a high-level description of the algorithm. Section 3 offers observations on the performance of the algorithm, and its potential for extension beyond telephone band speech. Appendix A provides a detailed description of the algorithm. This appendix was taken from Section 7 of the Draft ANSI T1 Standard on Multimedia Communications Delay, Synchronization, and Frame Rate Measurement, T1A1.5/96-101R8. The final page of Appendix A provides a block diagram of the algorithm.

2. Application and Overview

The algorithm was developed at ITS as a component of the ITS research on perception-based objective audio quality assessment tools. One desired output from this research is a perception-based algorithm that compares the contents of two digital audio files in a way that is consistent with human perception and judgment. Figure 1 shows how these two files are created and where the delay estimation algorithm is used. One of the files contains digital samples of a reference audio signal that went into the audio system under test. The other file contains digital samples of the audio signal as it came out of the audio system under test. These two files are referred to as the reference file and the test file, respectively. Because the audio system under test is a physical device, audio events can emerge from its output only after they have been presented to its input. The time required for the audio output to react to an input audio event is the delay of that audio system. If like segments of the reference and test files are to be compared by an objective audio quality assessment tool, the delay of the audio system under test must be removed before the comparison is made. Before the delay can be removed, it must be estimated. The algorithm described here estimates the delay of the contents of the test file relative to the contents of the reference file. It was designed for use on 4 kHz bandwidth speech, sampled at a rate of 8000 samples/second. Its potential for other uses is discussed in Section 3.

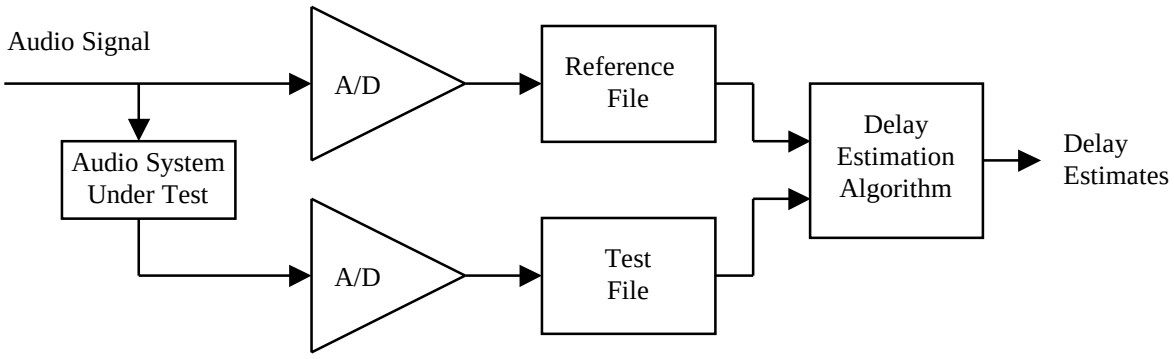


Figure 1. Example application of delay estimation algorithm

The algorithm features a coarse stage that uses speech envelopes, and a fine stage that uses speech power spectral densities. The coarse stage estimates delay to the nearest 4 ms. Whenever possible, the fine stage then refines this estimate to the nearest sample (125 μs when the sample rate is 8000 samples per second). Each stage involves a search over a range of possible delay values. The two stage process is efficient because the coarse stage can search a wide range of delay values, but at low resolution. If that same range were searched at high resolution, many more computations would be required. Once the coarse stage has finished its work, its low-resolution estimate can often be refined to a high resolution estimate by the fine stage that follows. The fine stage needs to search only a narrow range of delay values, consistent with the uncertainty of the coarse estimate.

Many digital speech coders do not preserve the speech waveform. By preserving a higher level statistical description of the speech waveform, but not the speech waveform itself, these devices achieve substantial compression ratios, and still generate output speech that sounds very similar to the input speech. When speech waveforms are not preserved, waveform cross-correlation and other waveform matching techniques will give ambiguous or erroneous delay estimates. For this reason, the coarse stage of the ITS algorithm uses speech envelopes, which are preserved by digital speech coders. Likewise, the fine stage uses speech power spectral densities (psd's) which are usually approximately preserved. For some speech coders and other speech devices, psd's are not adequately preserved and fine estimates are not possible. In other cases, multiple fine estimates will give inconsistent results. This indicates that, in the high resolution view at least, the delay is not constant. In these situations the coarse delay estimate, along with its inherent uncertainty, becomes the total delay estimate. Appendix A provides a detailed description of the algorithm. This appendix was excerpted from the Draft ANSI T1 Standard on Multimedia Communications Delay, Synchronization, and Frame Rate Measurement, T1A1.5/96-101R8. The final page of Appendix A provides a block diagram of the algorithm.

Finally, note that the timing relationship between the two A/D converters in Figure 1 must be known if an absolute delay value is required. While only positive delays are physically possible, the samples in the test file may lead or lag the samples in the reference file, depending on when the two A/D converters were started. If their starts were synchronized, then the samples in the test file must lag the samples in the reference file.

3. Observations on the Algorithm

Our experience with this algorithm is limited to 4 kHz bandwidth speech with a sample rate of f_s=8000 samples/second. A simple experiment shows that, under laboratory conditions, the delay estimation algorithm performs as we would expect. This experiment uses 30 speech files, each containing a distinct English language sentence pair. Three female and three male talkers were used, and each spoke five sentence pairs.

The 30 files were passed through the Modulated Noise Reference Unit (MNRU) at 6 different settings. The MNRU is a reference condition that adds amplitude correlated noise to a speech

signal[1]. This allows the MNRU to simulate the quantization noise that is produced by many waveform coders. The MNRU has a single control variable called *Q*, which essentially specifies the SNR of the output speech in dB. Our experiment uses *Q* = 0, 10, 20, 30, 40, and 50. Table 1 shows the mean and standard deviation (calculated across the 30 files) of the total estimated delay through the MNRU as a function of *Q*. Table 1 also indicates the percentage of the 30 files for which the fine stage was able to produce a useable delay estimate.

MNRU Setting	Mean of Estimated Delays	Standard Deviation of Estimated Delays	Percentage of Files with Useable Fine Estimate
Q = 0 dB	0.2 Samples	0.9 Samples	3%
10	1.1	1.6	97
20	1.5	0.7	100
30	1.6	0.6	100
40	1.7	0.7	100
50	1.7	0.6	100

Table 1. Delay Estimation Results for MNRU

Table 1 shows that when the speech waveform is severely distorted (*Q*=0 dB) the fine stage rarely produces a useable estimate. When the speech waveform is less distorted (*Q*≥20 dB) the fine stage always produces a useable estimate, and the standard deviations of the delay estimates are smaller than when the waveform is more distorted. Also, as the speech waveform becomes less distorted, the mean value of the estimated delay converges to 1.7 samples. This delay is caused by a filter inside the MNRU.

The 30 files were also passed through the T-Reference Condition (T-Ref) at 6 different settings. The T-Ref is a reference condition that removes and interpolates samples of digital speech signals to create a form of frequency modulation plus aliasing that can sound similar to some lower bit-rate coders[2]. The T-Ref has a single control variable called *T*, and smaller values of *T* correspond to larger amounts of distortion in the output speech signal. Our experiment uses *T* = 2, 4, 8, 16, 32, and 64. The theoretical, time-averaged delay of our software T-Ref implementation is $-\frac{2}{3}\cdot\frac{256}{T}$. Table 2 includes this theoretical delay value, as well as the mean and standard deviation (calculated across the 30 files) of the total estimated delay, and the percentage of the 30 files for which the fine stage was able to produce a useable delay estimate.

Table 2 reveals the following. The mean of the delay estimates track the theoretical delay values. The ability of the fine stage to produce a useable result increases as the waveforms become less distorted. The standard deviation of the delay estimates decreases as the waveforms become less distorted. Since the uncertainty of the coarse stage is ± 32 samples, the difference between the mean of the delay estimates and the theoretical delay value is

larger when the coarse stage operates alone, and is smaller when when the fine stage provides a refinement.

T-Ref Setting	Theoretical Average Delay	Mean of Estimated Delays	Standard Deviation of Estimated Delays	Percentage of Files with Useable Fine Estimate
T = 2	-85.3 Samples	-87.5 Samples	18.7 Samples	0%
4	-42.7	-37.3	12.1	7
8	-21.3	-30.9	5.9	33
16	-10.7	-8.2	7.9	53
32	-5.3	-5.8	2.6	87
64	-2.7	-3.0	0.9	97

Table 2. Delay Estimation Results for T-Reference

The algorithm also performs as we would expect under non-laboratory conditions. We have used the algorithm to estimate the delays of over 10,000 test speech files. These files include sentences and sentence pairs, and cover over 100 distinct combinations of speech coders and network conditions or reference conditions, 36 talkers and 3 languages. As an example of the selective behavior of the fine stage, we have observed that the fine stage of the algorithm produces a useful estimate all of the time when Adaptive Differential Pulse Code Modulation (ADPCM) coders are tested, some of the time when Vector Sum Excited Linear Predictive (VSELP) coders are tested, and never when Sinusoidal Transform Coders (STC's) are tested. The coarse stage always produces a useful estimate. Using either a priori information on coder delays, or a pair-wise listening technique, we have verified that the total delay estimates produced by the algorithm are correct, within the uncertainties inherent in the test files.

We have described an algorithm that generates a single delay estimate for a pair of files that were completely digitized before the algorithm was started. It is clear that this algorithm could also be implemented in an "on-line" or "live-data" fashion. Such an implementation would continuously read two streams of audio samples and generate a continuing stream of delay estimates.

We would expect this algorithm to perform similarly on wideband telephony speech (7 kHz bandwidth, f_s=16,000 samples/second.) We would expect performance to be much worse on more general audio signals. In particular, music can exhibit periodicities at many different time scales, ranging from milliseconds to 10's of seconds. These intrinsic periodicities can make delay estimation an extremely challenging task. A thorough study of a wide range of general audio signals might yield modifications to this algorithm that would provide maximal robustness against this intrinsic challenge.

4. Summary

We have presented a description of an algorithm for estimating the delay of telephony band speech. The algorithm features a coarse stage that uses speech envelopes, and a fine stage that uses speech power spectral densities. This approach is motivated by the need for delay estimates for audio and speech devices that do not preserve waveforms. The algorithm behaves as desired for speech signals that have been distorted by the MNRU, the T-Ref, and combinations of real speech coders and networks. The coarse stage of the algorithm always produces a useful estimate with an uncertainty of ± 32 samples (4 ms). The fine stage of the algorithm produces a refinement to that estimate whenever sufficient spectral detail is preserved to make such a refinement meaningful.

References

[1] ITU-T Recommendation P.810, "Modulated Noise Reference Unit," Geneva 1995.

[2] B. Cotton, "New Reference Condition for Very Low Bit Rate Voice Coder Evaluation" Proc. IEEE Global Telecommunications Conference, Orlando, 1992.

Appendix A

Section 7 of the Draft ANSI T1 Standard on Multimedia Communications Delay, Synchronization, and Frame Rate Measurement, T1A1.5/96-101R8.

7. Audio Measurements

7.1 Collecting Audio Frames for Measurements

7.1.1 Description of Audio Frames

An Audio Frame is a group of consecutive audio samples. The preferred number of samples in an Audio Frame depends on the audio sample rate, and is given in Section 5.
(Note: This chart suggests that an Audio Frame should contain 128 samples when the sample rate is 8000 samples/sec., 256 samples when the sample rate is 16,000 samples/sec., and 512 samples when the sample rate is 32,000, 44,100, or 48,000 samples/sec.)

7.1.2 Analog to Digital Conversion

The methods of measurement described in subsequent sections require the digitization of the analog audio signal. The digitization process results in audio samples, which can then be grouped into Audio Frames. The audio sample rate is determined by the bandwidth required for the subsequent measurements. Using the Nyquist theorem, the sampling rate must be at least twice this measurement bandwidth. For audio signals that are limited to speech, a sample rate of 8000 samples per second is sufficient. For other audio signals, higher sample rates may be required. The digitization process should result in at least 8 bits of precision. Depending on the signal-to-noise ratio of the audio signal, additional precision up to 16 bits will often benefit the methods of measurement that follow. The digitization process must include appropriate low-pass filtering to prevent aliasing, and care should be taken to match the impedance and balance of the audio signals.

7.1.3 Time Stamp Assignment

The time, T(n) associated with the Audio Frame n shall be read immediately following the digitization of the last sample in Audio Frame n, and before the next sample is digitized.

7.2 Delay Measurement for Audio

Many channels of potential interest are capable of delivering useable audio signals without preserving audio waveforms from input to output. This means that a robust delay measurement must not rely on audio waveforms alone. The measurement described here features a coarse stage that uses audio envelopes, and a fine stage that uses audio power spectral densities (PSD's). Audio envelopes and audio PSD's are approximately preserved by most channels.

The two stage process is efficient because the coarse stage searches a wide range of potential delay values, but at low resolution. If that same range were searched at high resolution, many more computations would be required. Once the coarse stage has finished its work, its low-resolution measurement can often be refined to a high resolution measurement by the fine stage that follows. The fine stage needs to search only a narrow range of potential delay

values, consistent with the uncertainty of the coarse measurement. For some channels, audio PSD's are not adequately preserved and fine measurements are not possible. In other cases, multiple fine measurements will give inconsistent results. In these situations the coarse delay measurement, along with its inherent uncertainty, becomes the final delay measurement.

7.2.1 General

A series of Audio Frames must be gathered from the channel input and the channel output before measurement can proceed. The use of more Audio Frames increases both the reliability and the complexity of the measurement. If a group of Audio Frames contains only the silence between words or phrases in a spoken conversation, no reliable measurement will be possible, and additional Audio Frames must be acquired before measurement can proceed. To detect an insufficient audio level condition, the RMS level of the audio samples in the group of Audio Frames acquired from the channel input should be compared with the nominal RMS level of the channel input. This nominal level may be taken from the channel input specifications or it may be measured. If the RMS level of the samples in the group of acquired channel input Audio Frames is more than 30 dB below the nominal channel input level, then additional Audio Frames must be acquired before the measurement can proceed. Similarly, the RMS level of the audio samples in the group of Audio Frames acquired from the channel output should be compared with the nominal RMS level of the channel output. If the RMS level of the samples in the group of acquired channel output Audio Frames is more than 30 dB below the nominal channel output level, then additional Audio Frames must be acquired before the measurement can proceed

For typical speech signals, the use of larger groups of Audio Frames reduces the possibility that the group contains only silence. Beyond this consideration, more Audio Frames bring more data to the measurement, and the measurement will be more reliable. For audio signals that are limited to speech, it is preferred that 256 Audio Frames be taken from the channel input and the channel output. The measurement will also work with 128 or 64 frames. When the sample rate is 8000 samples per second, and each frame contains 128 samples, these choices correspond to approximately 4 seconds, 2 seconds, or 1 second of speech signal respectively. The expected audio delay should not be more than 25% of the duration of the speech signal used in this measurement. When 256 frames (4 seconds) of speech signal are used, delays up to 1 second may be measured. When 64 frames (1 second) are used, only 250 ms of delay should be measured. The measurements are most efficiently computed when the number of frames acquired is a power of two.

7.2.2 Signal Preparation

When a measurement of audio delay is required, the Audio Frame most recently acquired from the channel input A(n), is concatenated with some number of previously acquired Audio Frames (e.g. A(n), A(n-1), ... A(n-255)), to form the most recent time-history of channel input samples. Similarly, the Audio Frame most recently acquired from the channel output A(m), is concatenated with the same number of previously acquired Audio Frames (e.g. A'(m), A'(m-1), ... A'(m-255)), to form the most recent time-history of channel output samples. According to Section 5, the time difference between these two acquisition

processes, is T'(m)-T(n). Positive values indicate that acquisition at the channel output happens after acquisition at the channel input.

The input samples are placed in an array called *ref*, which contains samples *ref(1), ref(2), ... ref(L1).* The output samples are placed in an identically sized array called *test*, which contains samples *test(1), test(2), ... test(L1).* The mean value of each array is then removed in order to eliminate any DC component in the digitized audio signals:

$$ref(i) = ref(i) - \tfrac{1}{L1} \cdot \sum_{j=1}^{L1} ref(j), \qquad test(i) = test(i) - \tfrac{1}{L1} \cdot \sum_{j=1}^{L1} test(j), \qquad 1 \le i \le L1.$$

Next, each array is normalized to a common level:

$$ref(i) = ref(i) \cdot \left[\tfrac{1}{L1-1} \sum_{j=1}^{L1} ref(j)^2, \quad \right]^{-\frac{1}{2}}, \quad test(i) = test(i) \cdot \left[\tfrac{1}{L1-1} \sum_{j=1}^{L1} test(j)^2, \quad \right]^{-\frac{1}{2}},$$

$1 \le i \le L1.$

7.2.3 Coarse Stage

The measurement methodology starts with a coarse stage that extracts and cross-correlates audio envelopes. Audio envelopes are approximately preserved by most channels.

7.2.3.1 Envelope Extraction

Audio envelopes are calculated as follows. The digitized audio signals in *ref* and *test* are rectified by taking the absolute value of each sample. Because the original digitized audio signals in *ref* and *test* will be required by the fine sage, the rectified signals, and other subsequent intermediate results are stored in the temporary arrays *ref_temp* and *test_temp:*

$$ref\_temp(i) = |ref(i)|, \quad test\_temp(i) = |test(i)|, \quad 1 \le i \le L1.$$

The rectified signals are then low-pass filtered to create audio envelopes with a bandwidth of approximately 125 Hz. It is this low-pass filtering and subsequent subsampling that gives the coarse stage its reduced resolution and reduced computational load. The bandwidth reduction factor and the subsampling factor are both specified by the variable *B.* Appropriate values of *B* for some common audio sample rates are given in table 7.1

Audio Sample Rate (samples/second)	B
8000	**32**
16,000	**64**
32,000	**128**
44,100	**176**
48,000	**192**

Table 7.1: Values of the bandwidth reduction factor, B.

When the audio sample rate is 8000 samples per second, the bandwidth must be reduced by a factor of *B*=32, from a nominal value of 4000 Hz to a nominal value of 125 Hz. The required bandwidth reduction can be adequately approximated using a seventh order, Infinite Impulse Response (IIR), low-pass Butterworth filter with a -3 dB point at 125 Hz:

$$out(i) = \sum_{j=0}^{7} b_j \cdot in(i-j) - \sum_{j=1}^{7} a_j \cdot out(i-j), \quad 1 \le i \le L1 ,$$

$$where \; out(i) = in(i) = 0 , \; i \le 0 .$$

The filter coefficients are given in table 7.2.

j	a_j	b_j
0	1.00000000	$0.00553833 \times 10^{-7}$
1	-6.55883158	$0.03876830 \times 10^{-7}$
2	18.44954612	$0.11630512 \times 10^{-7}$
3	-28.85178274	$0.19384125 \times 10^{-7}$
4	27.08958968	$0.19384206 \times 10^{-7}$
5	-15.27097592	$0.11630465 \times 10^{-7}$
6	4.78557610	$0.03876843 \times 10^{-7}$
7	-0.64312159	$0.00553831 \times 10^{-7}$

Table 7.2: Coefficient values for seventh order, IIR, low-pass Butterworth filter.

Both the *ref _temp* and *test _temp* arrays are low-pass filtered using this filter. Next *ref _temp* and *test _temp* are subsampled by retaining only every B^{th} sample, resulting in a total of *L2* samples. For example, when B=32, samples 1, 33, 65, etc. would be retained. When 256 Audio Frames, each with 128 samples are used as input to the coarse stage, *L1*=32,768, and *L2=L1*/*B*=1024 samples result from the subsampling process. Both *ref _temp* and *test _temp* now contain audio envelopes. Finally, the audio envelopes in *ref _temp* and *test _temp* are normalized. The mean value of each array is removed, and each array is divided by its standard deviation.

$$ref\_temp(i) = ref\_temp(i) - \tfrac{1}{L2} \cdot \sum_{j=1}^{L2} ref\_temp(j),$$

$$test\_temp(i) = test\_temp(i) - \tfrac{1}{L2} \cdot \sum_{j=1}^{L2} test\_temp(j),$$

$$ref\_temp(i) = ref\_temp(i) \cdot \left[\tfrac{1}{L2-1} \sum_{j=1}^{L2} ref\_temp(j)^2, \quad \right]^{-\frac{1}{2}},$$

$$test\_temp(i) = test\_temp(i) \cdot \left[\tfrac{1}{L2-1} \sum_{j=1}^{L2} test\_temp(j)^2, \quad \right]^{-\frac{1}{2}},$$

$1 \le i \le L2$.

7.2.3.2 Envelope Cross-Correlation

The cross-correlation between the audio envelopes in *ref _temp* and *test _temp* is calculated by way of a circular convolution, which in turn is calculated by way of Discrete Fourier Transforms (DFT's) or Fast Fourier Transforms (FFT's). First, the array *ref _temp* is extend from length *L2* to length $2 \cdot L2$ by appending *L2* zeros. In the example above, *L2*=1024 zeros would be added to arrive at a final array size of 2048. Next the array *test _temp* is time-reversed. To do this in-place, samples 1 and *L2* of *test _temp* are exchanged, as are samples 2 and *L2-1*, samples 3 and *L2-2*, etc. When *L2* is even, the final exchange is between samples *L2*/2 and *L2*/2 + 1. When L2 is odd, the final exchange is between samples *L2*/2 - 1/2, and *L2*/2 + 3/2. After this time reversal, *test _temp* is extended from length *L2* to length $2 \cdot L2$ by appending *L2* zeros.

Now *ref _temp* and *test _temp* are transformed using DFT's or FFT's. When the array length, $2 \cdot L2$, is a power of two, FFT's can be used. If $2 \cdot L2$ is not a power of two, DFT's can be used. As an alternative, the number of zeros appended in the previous step may be increased so that the array length is a power of two and FFT's may then be used. In any case, an in-place transformation algorithm may be used, resulting in transformed versions of *ref _temp* and *test _temp* overwriting the previous versions. The transformations result in complex numbers.

Next, the complex samples stored in *ref _temp* and *test _temp* are multiplied, sample by sample, and the complex results go into an new array called *cross_corr*, which has the same length as *ref _temp* and *test_temp*:

$$cross\_corr(i) = ref\_temp(i) \cdot test\_temp(i), \text{ for } i = 1 \text{ to } 2 \cdot L2 \ .$$

The array *cross_corr* is now Inverse Fast Fourier Transformed or Inverse Discrete Fourier Transformed, as dictated by its length. An in-place transformation may be used. In theory, the resulting contents of *cross_corr* would be real numbers. In practice, finite-precision

calculations yield a small imaginary component. At this point, the real part of *cross_corr* is retained and the imaginary part is discarded. Next, each result in *cross_corr* is normalized:

$$cross\_corr(i) = cross\_corr(i) / (L2 - 1),\ 1 \le i \le 2 \cdot L2\ .$$

Note that this normalization is required in order to get true cross-correlation values between -1 and +1, but it does not affect the smoothing or peak-finding steps that follow.

The array *cross_corr* holds the values of the cross-correlation's between the speech envelopes in *ref_temp* and *test_test* at every possible shift of those envelopes. These results are then smoothed with a symmetric, second-order, low-pass FIR filter, and stored in a smoothed cross-correlation array:

$$cross\_corr\_s(i) = .25 \cdot cross\_corr(i-1) + .5 \cdot cross\_corr(i) + .25 \cdot cross\_corr(i+1)\ ,$$
$$2 \le i \le 2 \cdot L2 - 1,$$
$$cross\_corr\_s(i) = cross\_corr(i),\ i = 1,\ 2 \cdot L2.$$

After this smoothing, the largest value in *cross_corr_s* is taken as an indication of the coarse delay:

$$coarse\_delay = (L2 - j) \cdot B \text{ samples},$$
$$= \frac{(L2 - j) \cdot B}{sample\_rate} \text{ seconds},$$

where $cross\_corr\_s(j) > cross\_corr\_s(i),\ 1 \le i \le 2 \cdot L2,\ i \ne j$.

The uncertainty in the value of *coarse_delay* at this point is taken to be ± *B* samples. If *cross_corr_s* does not contain a unique maximal value, then the measurement must be made again using new audio samples.

7.2.4 Fine Stage

In many cases, the ±*B* sample uncertainty inherent in the coarse measurement of audio delay can be reduced by a fine stage of the delay measurement.

7.2.4.1 Location Selection

The fine stage is performed at *n1* locations in the acquired audio signal. When audio signals are limited to speech and 256 Audio Frames are used in the measurement of audio delay, the value of *n1* is 6. Other values of n1 may be more appropriate for other audio signals. At each location, a range of potential delay values from -3·*B* to 3·*B* samples is searched.

The locations where the fine stage is performed are randomly selected. At each location, 8·*B* samples are taken from the array *ref* and are stored in *ref_temp* and 2·*B* samples are taken

from the array *test* and are stored in *test_temp*. The samples taken from *test* are offset by the measured coarse delay:

ref_temp(*i*) = *ref*(loc*ation-4·B-1+i), 1≤ i ≤8·B* ,
test_temp(*i*) = *test(location+coarse_delay-B-1+i), 1≤ i ≤2·B* ,

where *location* is a uniformly distributed pseudo-random variable from the interval:
[max(4·*B* +1 , 1-coarse_delay+*B*) , min(*L1*-4·*B* +1 , *L1*-coarse_delay-*B*+1)].

The fine delay measurement will not work in silent regions. Two level tests are conducted at each location to insure that the audio signal there is within 30 dB of the average audio signal level:

$$-30 \le 10 \cdot \log_{10}\left[\frac{1}{8 \cdot \mathbf{B}-1} \sum_{i=1}^{8 \cdot \mathbf{B}} ref\_temp(i)^2 \right], \quad -30 \le 10 \cdot \log_{10}\left[\frac{1}{2 \cdot \mathbf{B}-1} \sum_{i=1}^{2 \cdot \mathbf{B}} test\_temp(i)^2 \right].$$

If either of the level tests is failed, then a new location must be selected.

7.2.4.2 Power Spectral Density Calculations

The fine stage works by cross-correlating audio power spectral densities (PSD's) at each of the selected locations. The PSD's are calculated as follows. The 8·*B* samples in *ref_temp* are broken into groups of 2·*B* samples per group. There are 6·*B*+1 such groups. Each group of samples is stored in an array called ref_temp_i:

ref_temp_i(*j*)=*ref_temp*(*i+j-1*) , 1≤ i ≤ 6·*B*+1, 1≤ j ≤2·*B* .

Each *ref_temp_i* array and the *test_temp* array is multiplied by a Hamming window, and then transformed to the frequency domain using a length 2·*B* DFT or FFT. These steps can be done in place:

ref_temp_i(*j*) = *ref_temp_i*(*j*) · {.54 - .46 · cos(2π(j-1)/(2·*B-1)*)}, 1≤ i ≤6·*B*+1, 1≤ j ≤2·*B* ,
test_temp(*j*) = *test_temp*(*j*) · {.54 - .46 · cos(2π(j-1)/ (2·*B-1)*)}, 1≤ j ≤2·*B* ,
ref_temp_i = FFT (*ref_temp_i*) , 1≤ i ≤6·*B*+1 ,
test_temp = FFT (*test_temp*) .
In the frequency domain, only the first *B*+1 complex samples in each array are unique, so only those samples are saved. The magnitude of each retained sample is taken, resulting in the square root of the power spectral density of each frame. These results are referred to as PSD's for simplicity.

ref_temp_i(*j*) = | *ref_temp_i*(*j*) | , 1≤ i ≤6·*B*+1, 1≤ j≤*B*+1 ,
test_temp(*j*) = | *test_temp*(*j*) | , 1≤ j ≤ *B*+1 .

The mean value of each PSD is then removed:

$$ref\_temp\_i(j) = ref\_temp\_i(j) - \tfrac{1}{B+1} \cdot \sum_{j=1}^{B+1} ref\_temp\_i(j),\ 1 \le i \le 6 \cdot B+1,\ 1 \le j \le B+1,$$

$$test\_temp(j) = test\_temp(j) - \tfrac{1}{B+1} \cdot \sum_{j=1}^{B+1} test\_temp(j),\ 1 \le j \le B+1.$$

7.2.4.3 Power Spectral Density Cross-Correlation

A cross-correlation value is calculated between the PSD stored in the *test_temp* array and each of the 6·*B*+1 PSD'S stored in the *ref_temp_i* arrays.

$$cross\_corr(i) = \frac{\left(\sum_{j=1}^{B+1} ref\_temp\_i(j) \cdot test\_temp(j) \right)}{\left(\sum_{j=1}^{B+1} ref\_temp\_i(j)^2 \right)^{\frac{1}{2}} \left(\sum_{j=1}^{B+1} test\_temp(j)^2 \right)^{\frac{1}{2}}},\ 1 \le i \le 6 \cdot B+1.$$

The array *cross_corr* now holds the values of the cross-correlations between the reference and test PSD's at each time-domain shift. Note that the second term in the denominator of the equation for *cross_corr* is a normalizing constant that is required to get true cross-correlation values between -1 and +1. It does not have any impact on the peak-finding that follows, but does impact subsequent processing of the fine delay measurements. The largest value in *cross_corr* is taken as an indication of the fine delay:

$$fine\_delay\_k = (3 \cdot B+1) - j \text{ samples},$$
$$= \frac{(3 \cdot B+1) - j}{sample\_rate} \text{ seconds},$$
$$corr\_k = cross\_corr(j),\ 1 \le k \le n1,$$
$$\text{where } cross\_corr(j) > cross\_corr(i),\ 1 \le i \le 6 \cdot B+1.,\ i \ne j.$$

If *cross_corr* does not contain a unique maximal value, then the fine stage procedure must be repeated at a new location. This entire fine stage, starting with the selection of a location, is repeated *n1* times, resulting in *n1* fine delay measurements stored in *fine_delay_1, fine_delay_2, ... fine_delay_n1,* and *n1* corresponding correlation values stored in *corr_1, corr_2, ... corr_n1*, respectively. Note that each of the fine delay estimates will fall between -3·*B* and 3·*B*, inclusive.

7.2.4.4 Fine Delay Measurement Processing

Once the *n1* fine delay measurements and corresponding cross-correlation values have been calculated, they are further processed to determine how they should be used.

First, each of the *n1* correlation values are tested against a threshold:

$$\sqrt{\tfrac{1}{2}} \le corr\_k \Rightarrow fine\_delay\_k \text{ is retained}, 1 \le k \le n1.$$

By this process, only fine delay measurements where at least half the PSD variance is accounted for are retained. The number of fine delay measurements that pass this test is *n2*, and the measurements are now renumbered as *fine_delay_1*, *fine_delay_2*, ... *fine_delay_n2*. If *n2* < *n1*/2, the fine stage will not produce a useful result. In this event, the value of *fine_delay* is set to "invalid" and the fine stage is terminated.

If *n2* ≥ *n1*/2, the fine stage continues and tests the remaining *n2* fine delay measurements for consistency with the coarse delay measurement. Since the uncertainty in the coarse delay measurement is ±*B* samples, and the coarse delay has been removed, only fine delay measurements between -*B* and *B* samples are retained:

$$\left| fine\_delay\_k \right| \le B \Rightarrow \quad fine\_delay\_k \text{ is retained}, 1 \le k \le n2.$$

The number of fine delay measurements that pass this test is *n3*, and the measurements are now renumbered as *fine_delay_1*, *fine_delay_2*, ... *fine_delay_n3*. If *n3* < *n1*/2, the fine stage will not produce a useful result. In this event, the value of *fine_delay* is set to "invalid" and the fine stage is terminated.

If *n3* ≥ *n1*/2, the fine stage continues and tests the for consistency among the remaining *n3* fine delay measurements. This test requires a search through all possible subsets of size *n3*, *n3*-1, on down to size *n1*/2. There is one possible subset of size *n3*, *n3*-1 possible subsets of size *n3*-1, *n3*·(*n3*-1)/2 possible subsets of size *n3*-2, and so forth. For each subset, the spread of the fine delay measurements is tested:

$$\max_i\{fine\_delay\_i\} - \min_i\{fine\_delay\_i\} \le \tfrac{B}{2}, \quad fine\_delay\_i \in \text{current subset}.$$

The largest subset that passes this test is called the final subset. The fine stage fails to produce a useful result when:

- no subset passes this test, or
- there is not a single, largest subset that passes this test.

In either of these events, the value of *fine_delay* is set to "invalid" and the fine stage is terminated.

The number of fine delay measurements in the final subset is *n4*. The mean value of these *n4* fine delay measurements is taken as the final fine delay measurement:

$$fine\_delay = \tfrac{1}{n4} \cdot \sum_{j=1}^{n4} fine\_delay\_i, \quad fine\_delay\_i \in \text{final subset}.$$

The spread of the *n4* measurements in the final sub-set is retained as a measure of uncertainty in the final fine delay measurement:

$$spread = \max_i\{fine\_delay\_i\} - \min_i\{fine\_delay\_i\}, \quad fine\_delay\_i \in \text{final subset}.$$

7.2.5 Combining Coarse and Fine Stage Results

If the fine stage was not able to produce a useful fine delay measurement, then the fine stage will have set *fine_delay* to "invalid". In this case, the coarse measurement alone becomes the delay measurement. If the fine stage was able to produce a useful fine delay measurement, then the coarse measurement is augmented by that fine measurement and the uncertainty is reduced from that of the coarse measurement alone:

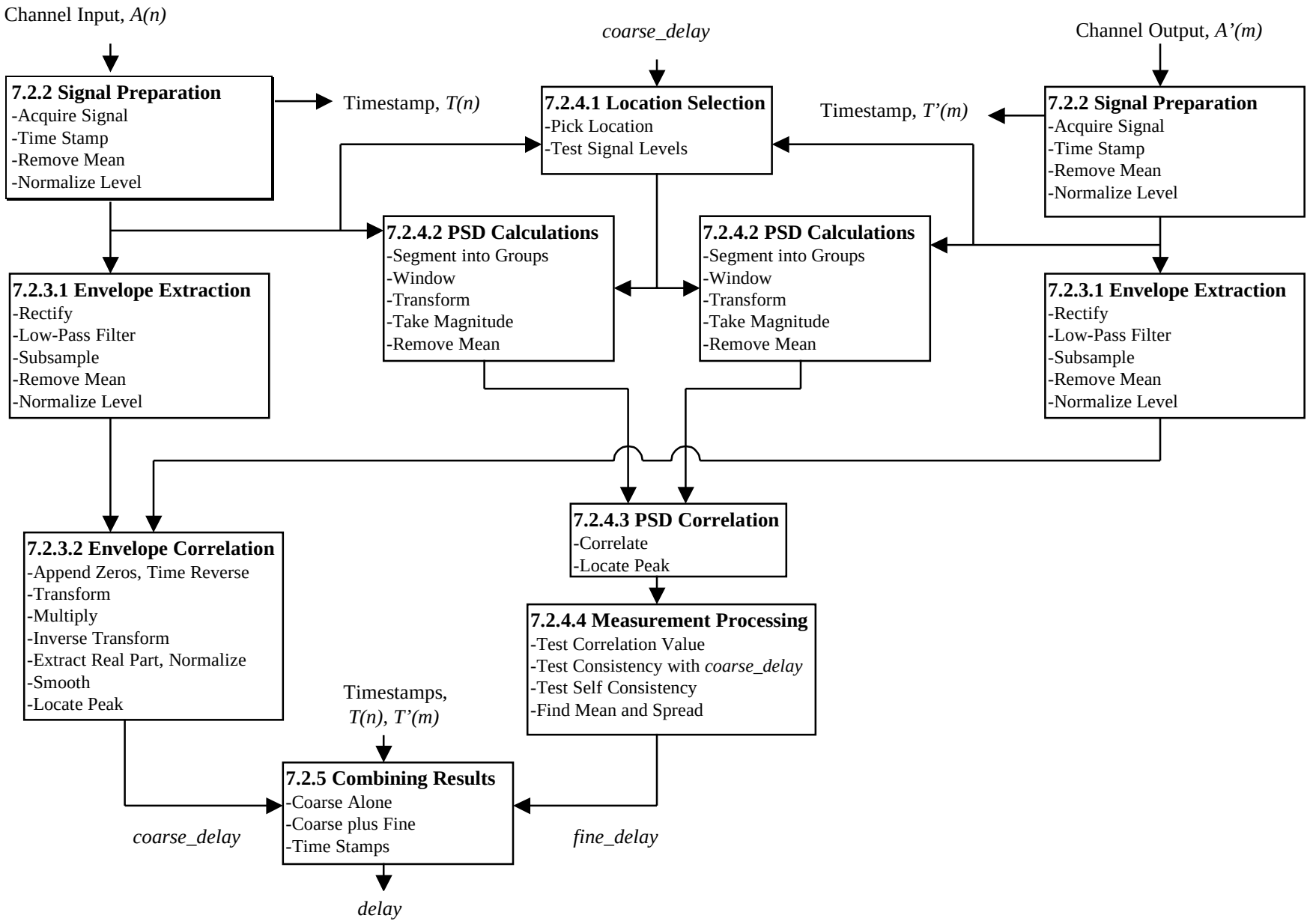
$$fine\_delay = \text{"}invalid\text{"} \Rightarrow delay = coarse\_delay \pm B,$$
$$fine\_delay \neq \text{"}invalid\text{"} \Rightarrow delay = coarse\_delay + fine\_delay \pm spread.$$

These values of *delay* are correct only when the acquisition of audio samples from channel input and the channel output are simultaneous. Time stamps can be used to correct the delay measurement for non-simultaneous acquisition:

$delay = delay + T'(m) - T(n)$, where T'(m) is the time associated with the start of sample acquistion at the channel output,and T(n) is the time associated with the start of sample acquistion at the channel input.

Variables used in Section 7.

B:	bandwidth reduction factor and subsampling factor
coarse_delay:	delay as measured by coarse stage
cross_corr:	temporary array, ultimately holds cross-correlation values
cross_corr(i):	i^{th} element of cross_corr array
cross_corr_s:	smoothed version of cross_corr in coarse stage
cross_corr_s(i):	i^{th} element of cross_corr_s array
delay:	final output of two-stage delay measurement
fine_delay:	delay as measured by fine stage
fine_delay_k:	k^{th} fine delay masurement
L1:	number of audio samples input to measurement
L2:	number of audio samples after subsampling
location:	location where fine stage makes a measurement
n1:	number of measurements made by fine stage
n2:	number of fine stage measurements retained after first test
n3:	number of fine stage measurements retained after second test
n4:	number of fine stage measurements retained after third test
ref:	array of audio samples from channel input
ref(i):	i^{th} element of ref array
ref_temp:	temporary storage array for channel input audio samples as they are processed
ref_temp(i):	i^{th} element of ref_temp array
ref_temp_i :	temporary storage array for channel input audio samples as they are processed
ref_temp_i(j):	j^{th} element of ref_temp_i array
sample_rate:	rate at which channel input and channel output are digitized
spread:	spread in the final subset of fine delay measurements
test:	array of audio samples from channel output
test(i):	i^{th} element of test array
test_temp:	temporary storage array for channel output audio samples as they are processed
test_temp(i):	i^{th} element of test_temp array


Channel Input, A(n)
coarse_delay
Channel Output, A'(m)
7.2.2 Signal Preparation
-Acquire Signal
-Time Stamp
-Remove Mean
-Normalize Level
Timestamp, T(n)
7.2.4.1 Location Selection
-Pick Location
-Test Signal Levels
Timestamp, T'(m)
7.2.2 Signal Preparation
-Acquire Signal
-Time Stamp
-Remove Mean
-Normalize Level
7.2.4.2 PSD Calculations
-Segment into Groups
-Window
-Transform
-Take Magnitude
-Remove Mean
7.2.4.2 PSD Calculations
-Segment into Groups
-Window
-Transform
-Take Magnitude
-Remove Mean
7.2.3.1 Envelope Extraction
-Rectify
-Low-Pass Filter
-Subsample
-Remove Mean
-Normalize Level
7.2.3.1 Envelope Extraction
-Rectify
-Low-Pass Filter
-Subsample
-Remove Mean
-Normalize Level
7.2.4.3 PSD Correlation
-Correlate
-Locate Peak
7.2.3.2 Envelope Correlation
-Append Zeros, Time Reverse
-Transform
-Multiply
-Inverse Transform
-Extract Real Part, Normalize
-Smooth
-Locate Peak
7.2.4.4 Measurement Processing
-Test Correlation Value
-Test Consistency with coarse_delay
-Test Self Consistency
-Find Mean and Spread
Timestamps,
T(n), T'(m)
7.2.5 Combining Results
-Coarse Alone
-Coarse plus Fine
-Time Stamps
coarse_delay
fine_delay
delay