ATIS STANDARD

ATIS-0100801.04.2005(R2015)

Multimedia Communications Delay, Synchronization, and Frame Rate

AMERICAN NATIONAL STANDARD FOR TELECOMMUNICATIONS

AMERICAN NATIONAL STANDARD

Notice of Disclaimer & Limitation of Liability

ATIS-0100801.04.2005(R2015), *Multimedia Communications Delay, Synchronization, and Frame Rate*

Is an American National Standard developed by the **Quality of Service and Reliability (QoSR)** Subcommittee under the **ATIS Packet Technologies and Systems Committee (PTSC)**.

Published by

Alliance for Telecommunications Industry Solutions
1200 G Street, NW, Suite 500
Washington, DC 20005

(R2015)

ATIS-0100801.04.2005

[Revision of T1.801.04.1997(R2002)]

American National Standard for Telecommunications

MULTIMEDIA COMMUNICATIONS DELAY, SYNCHRONIZATION, AND FRAME RATE

Secretariat

Alliance for Telecommunications Industry Solutions

Approved December 13, 2005

American National Standards Institute, Inc.

Abstract

This standard addresses delay and synchronization issues in Multimedia systems that may combine video, audio, and data channels. Video delay can vary widely over short sequences, audio and video sequences may be distorted during transmission, and data streams can have little or no structure and may contain bit errors. Although each media presents unique measurement challenges, the methods specified here meet and overcome them. This standard also gives considerations for joint measurements and specifications for internal clocks that provide time stamps. Performance standards development proceeds in three stages, specifying parameters, methods of measurement, and (usually numerical) limits for services, networks or equipment. This specification addresses the first two stages, laying a foundation for the third.

FOREWORD

The Alliance for Telecommunication Industry Solutions (ATIS) serves the public through improved understanding between carriers, customers, and manufacturers. The Network Performance, Reliability, and Quality of Service Committee (PRQC) -- formerly T1A1 -- develops and recommends standards, requirements, and technical reports related to the performance, reliability, and associated security aspects of communications networks, as well as the processing of voice, audio, data, image, and video signals, and their multimedia integration. PRQC also develops and recommends positions on, and foster consistency with, standards and related subjects under consideration in other North American and international standards bodies.

An aspect of true Multimedia Communications Systems that sets them apart from a mere collection of unrelated channels is their ability to maintain a temporal relationship between the different media. This standard specifies the parameters and measurement methods to assess relative synchronization between media channels, and two other key aspects of temporal quality. Transmission time, or delay through a channel, is critical when assessing a system's suitability for conversational and other interactive uses. Frame inter-arrival time and its reciprocal, frame rate, characterize a system's ability to deliver information continuously and consistently.

Today's Multimedia systems combine video, audio, and data channels to enhance communications. This standard covers all these media. Video delay can vary widely over short sequences, audio and video sequences may be distorted during transmission, and data streams can have little or no structure and may contain bit errors. Although each media presents unique measurement challenges, the methods specified here meet and overcome them. The Mean Square Error based method expects and measures instantaneous video delay variations if present, setting this approach apart from earlier methods. The audio delay method accommodates channels where the original speech waveform is not preserved and the delay may be time-varying. The data channel methods take advantage of native structures when possible and tolerate bit errors. All the methods allow test signals representative of the intended system application.

Performance standards development proceeds in three stages, specifying parameters, methods of measurement, and (usually numerical) limits for services, networks, or equipment. This specification addresses the first two stages, laying a foundation for the third.

This standard also gives considerations for joint measurements and specifications for internal clocks that provide time stamps. There is also a discussion of time stamp position in informative annex A. Annex B is a guide to the mathematical symbols used in the standard. For those who seek more information, a bibliography of related standards and publications may be found in Annex C. Annex D contains a full definition of the audio delay measurement algorithm described in clause 7, in the form of computer code.

This standard was originally developed by Technical Subcommittee T1A1 (now ATIS PRQC) under the Video Teleconference/Video Telephony Performance and Advanced Television Performance projects. The first project began in 1988 under the auspices of Technical Subcommittee T1Q1. The standard's most recent revision (2005) was developed under PRQC Issue A0024. The revised standard complements PRQC's recent specifications covering video test scenes, performance terms and definitions, and other objective measurements.

Suggestions for improvement of this standard will be welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee PRQC. Committee approval of this standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, the PRQC had the following members:

The Quality of Service (QoS) Task Force was responsible for the development of this document.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

American National Standard for Telecommunications –

Multimedia Communications Delay, Synchronization, and Frame Rate

1 SCOPE, PURPOSE, & APPLICATION

1.1 Scope

This standard covers test methodologies for multimedia transmission systems utilizing digital transport facilities. It gives a set of measurement parameters, without providing limits, to characterize the following aspects of system performance:

a) Active Video Frame inter-arrival time, which is the reciprocal of the elementary frame rate;
b) Visual channel transmission time, also called video delay;
c) Audio channel transmission time (or audio delay);
d) Data channel transmission time or delay (and frame inter-arrival time);
e) Temporal synchronization between channels.

This standard specifies methods of measurement for these parameters in the applications described in 1.3. The standard's scope is limited to cases where appropriate media input and output interfaces are present, or where these interfaces can be made available with optional test fixtures.

The following applications are beyond the scope of this standard:

a) *Measuring aspects of system performance other than delay, synchronization, and frame rate*. It is important to point out that delay, synchronization, and frame rate measurements do not completely characterize the quality of a multimedia transmission system. For example, the reproduction quality of Video Frames from input to output is also of obvious importance to users. Subjective test methods employing representative users yield the best available assessment of video transmission quality. The optimization of such subjective performance for all quality parameters may take precedence over the optimization of the results of parametric measurements performed according to this standard. The bibliography lists several international recommendations on subjective assessment methods for entertainment and teleconference quality video transmission systems. It also lists standards for assessment of audio and data channels.

b) *Unrestricted choice of useful and representative source content*. The methods of measurement specified here require restrictions on their source signals for testing. Video source sequences with high motion activity often cause increased delay, decreased frame rate, and skewed audiovisual synchronization in some multimedia applications. Therefore, measurements should use test scenes which are realistic for the application of the multimedia system under evaluation. Continuous audio tones or still video sequences will produce ambiguous delay measurements (but delay is of limited importance under these conditions). Other limitations are given in the sections for each measurement method.

c) *Measuring the performance aspects of systems where the input and output interfaces are not accessible*.

d) *Limits for the parameters*. This standard only provides methods to measure these parameters without providing values for evaluation.

1.2 Purpose

The purpose of this standard is to ensure the uniform application of, provide a framework for, and provide definitions of performance parameters for transmission systems utilizing digital transport facilities, consistent with the scope. This standard is intended to provide a common understanding among manufacturers, service providers, and their customers.

1.3 Application

1.3.1 User-to-User Channels

Ideally, the delay measurement would be conducted at the user interfaces, so as to characterize the entire user-to-user delay. The complete *user-to-user channel* begins and ends with user interface devices. For example, consider the visual channel with its camera and display components, as shown in figure 1.

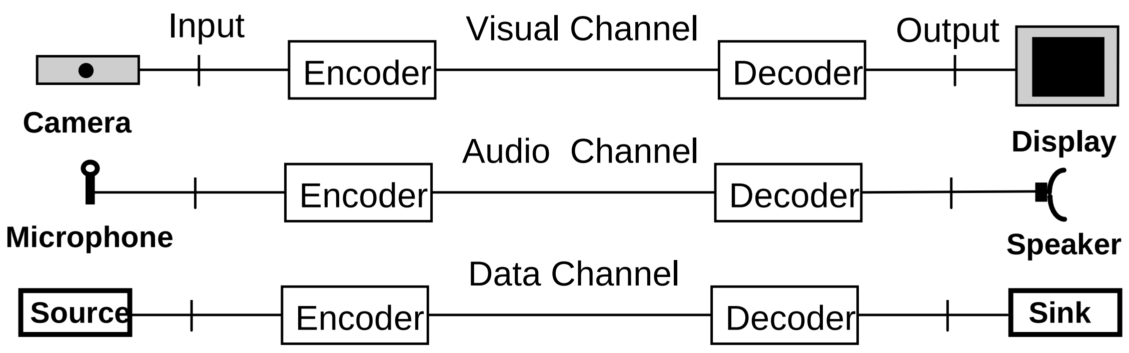

Figure 1 - User-to-User Channels in a Multimedia System

Unfortunately, signals enter and leave this channel in the form of light, making the generation and collection of suitable signals for measurement a difficult task. To simplify the interconnection of measurement equipment with the channel, the test channel between electrical interface connectors at the camera output and at the display input is specified. This has the advantage of providing more physical and logical structure to the test interface. The additional delay contributed by a camera and display could be assessed separately (these delays are expected to be constrained within the sample/display interval, may be constant for displays, and are usually test-signal independent) and added to the measurements of variable delay made in accordance with this standard.

Similar input and output interfaces in audio user-to-user channels and data user-to-user channels can be identified.

1.3.2 Applicable Configurations

The following channel configurations are appropriate applications of this standard. Each figure shows the necessary input and output interfaces.

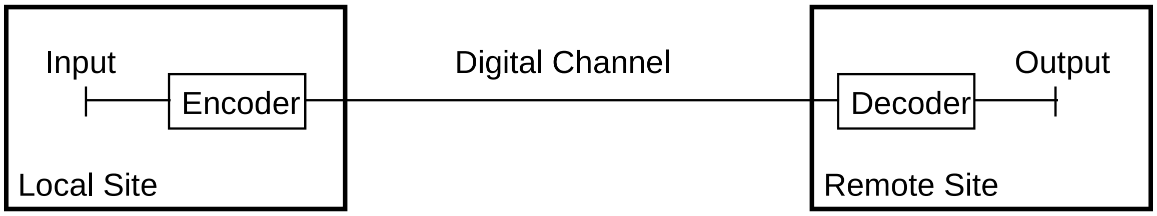

Figure 2 - End-to-End Measurement

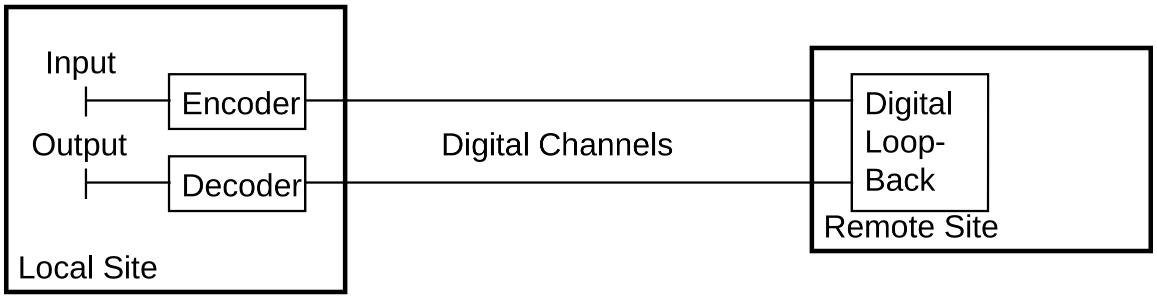

Figure 3 - Remote Digital Loop-Back Measurement

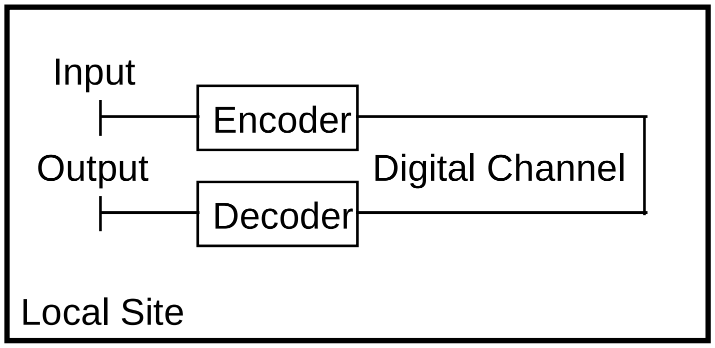

Figure 4 - Local System Measurement

These figures show only an Encoder, Decoder, and Digital Channel for simplicity. The components that may comprise the media channel in these tests are not strictly limited, and further examples may be found in the definitions of the channels (clause 3).

The results of measurements conducted in the configurations of Figures 3 and 4 can be used to assess the delay of the digital channels by removing the delay attributable to the local system.

Figure 5 shows a video channel measurement configuration with limited application. This would constitute a two-way delay measurement of two one-way systems and permit a single measurement device.

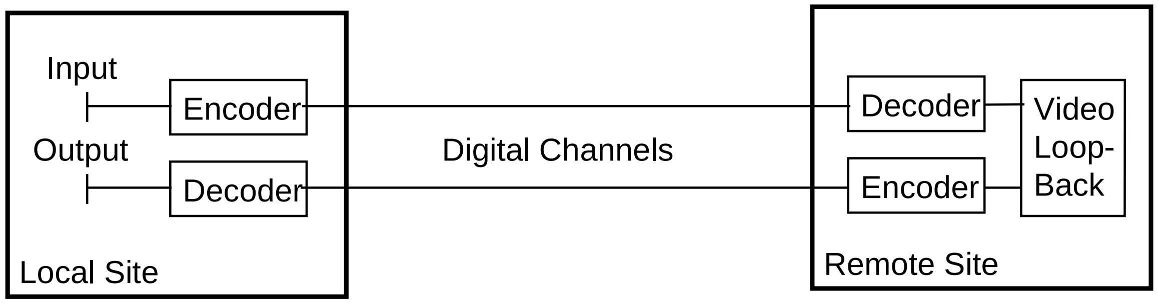

Figure 5 - Remote Video Loop-Back Measurement

The video loop-back is not appropriate for cases where the digital channels have asymmetrical delay. For video teleconference systems encoding less than 30 frames per second, the coding of the forward path may influence the transmission delay of the return path, which would make loop-back testing inappropriate in this case. Under these circumstances, the two-way measurement would not reveal the desired one-way assessment.

NOTE – One-way assessment for symmetrical systems is simply one-half of the two-way delay.

1.3.3 Applicable interfaces – Video

The work leading to the development of this standard was primarily conducted using composite analog video signal interfaces (conforming, at least in spirit, with SMPTE 170M). Many video transmission systems in use today, including video conference systems, have composite interfaces available. This is clearly a convenient interface for the measurements described in this standard.

However, the design of video conference systems is rapidly changing from a collection of components (using the composite interface) to more integrated systems. Furthermore, the demands of high quality video production exceed the capabilities of the composite analog signal. It will be necessary to apply this standard at new interfaces to keep pace with advancing technologies. Digital component interfaces, computer monitor RGB interfaces, and digital camera interfaces are likely candidates.

The measurement parameters defined here simply require the ability to supply Video Frames to the input and collect and compare Video Frames at the output of a visual channel. No technique demands a composite interface, per se.

To apply this standard at a general video input interface, equipment shall supply source frame pixel information according to the specific interface’s physical, logical, and timing conventions.

At a general video output interface, equipment shall be able to synchronize collection with the output information structure and identify the Video Frame area (for analog composite active video area, see 5.3 of T1.801.03-2003 (R2008), for safe action area and safe title area, see SMPTE RP 27.3).

To facilitate the measurements described in this standard on fully integrated systems, optional interface access features will be needed to support testing. The complexity of individual systems may not be appreciably increased if nearly all of the additional functions required to implement these interfaces are contained in the optional sub-system. These interface features might be useful in other activities, such as fault isolation and manufacturing quality assurance, and should be desirable to manufacturers on this basis.

1.3.4 Applicable interfaces – Audio

This specification is applicable at all standardized audio interfaces.

1.3.5 Applicable interfaces – Data

This specification is applicable at all standardized data interfaces.

2 NORMATIVE REFERENCES

The following standards contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

T1.504a-1991, *Packet-Switched Data Communication Service – Performance Measurement Methods.*[1]

T1.517-1995, *Digital Transport of Video Teleconferencing/Video Telephony Signals – Video Test Scenes for Subjective and Objective Performance Assessment.*[1]

T1.801.02-1996, *Digital Transport of Video Teleconferencing/Video Telephony Signals – Performance Terms, Definitions, and Examples.*[1]

T1.801.03-1996, *Digital Transport of One-Way Video Signals – Parameters for Objective Performance Assessment* [1]

ITU-T Recommendation BT.601-5, *Encoding Parameters of Digital Television for Studios.*[2]

SMPTE RP 27.3-1989, *Recommended Practice, Specifications for Safe Title Areas, Test Pattern for Television Systems.*[3]

SMPTE 125M-1992, *SMPTE Standard for Television – Component Video Signal 4:2:2 – Bit-Parallel Digital Interface.*[3]

SMPTE 259M-1993, *SMPTE Standard for Television – 10-Bit 4:2:2 Component and 4f NTSC Composite Signals – Bit-Parallel Digital Interface.*[3]

SMPTE 170M-1994, *SMPTE Standard for Television – Composite Analog Video Signal – NTSC for Studio Applications.*[3]

[1] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS), 1200 G Street N.W., Suite 500, Washington, DC 20005. < https://www.atis.org/docstore/default.aspx >

[2] This document is available from the International Telecommunications Union. < http://www.itu.int/ITU-T/ >

[3] This document is available at < http://store.smpte.org >.

3 Definitions, Acronyms, & Abbreviations

3.1 Definitions

3.1.1 Multimedia Communication System: A system that handles more than one media stream in a synchronized way from the user's point of view. The system may allow interconnection of multiple parties, multiple connections, and the addition or deletion of resources and users within a single communication session.

3.1.2 Media Stream: A sequence of presentation units intended to convey some specific content.

3.1.3 Coding Hierarchy Levels: The nested units of signal representation into which a media stream can be decomposed.

3.1.4 Content Hierarchy Levels: The nested units of information into which a media stream can be decomposed.

3.1.5 Presentation Unit: The smallest convenient division of a media stream (defined by the measurement system) that conveys an independent, self-contained unit of content, from among the content hierarchy levels present in the stream.

3.1.6 Video Frame: A presentation unit of the visual channel. The lowest level of the content hierarchy in a video media stream, where differences between sequential units appear throughout the unit of presentation. The content hierarchy of this standard may re-use the terms in some coding hierarchies, if necessary. For NTSC interfaces, a Video Frame is defined as one NTSC Field, where an NTSC Field is specified in SMPTE 170M.

3.1.7 Audio Frame: A presentation unit of the audio channel. 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 clause 5. These Audio Frames have no relationship to the frames designated by certain audio/speech codecs.

3.1.8 Data Frame: A presentation unit of the data channel. A group of consecutive data bits. The preferred number of bits in a Data Frame depends on the application for the data channel.

3.1.9 Digital Channel: A means for conveying information from one point to another in digital form. A digital channel may be implemented on a network composed of digital communications components.

3.1.10 Visual Channel: A means for delivering Video Frames from one point to another. A sequence of frames submitted to the channel input results in a similar (not necessarily identical) sequence of frames at the channel output. The visual channel may be comprised of the following components: video format conversion devices, encoders (compressors) and decoders (decompressors), rate smoothing buffers, multiplexors and demultiplexors, modulators and demodulators, transmission facilities, switches, multi-point conference units, and other components necessary to achieve the desired channel characteristics.

3.1.11 Audio Channel: A means for delivering audio signals from one point to another. An audio waveform submitted to the channel input results in a similar (not necessarily identical) waveform at the channel output. The audio channel may be comprised of the following components: encoders (compressors) and decoders (decompressors), buffers, multiplexors and demultiplexors, modulators and demodulators, transmission facilities, switches, multi-point conference units, and other components necessary to achieve the desired channel characteristics.

3.1.12 Data Channel: A means for delivering data from one point to another. A sequence of data bits submitted to the channel input results in a similar (not necessarily identical) sequence of bits at the channel output. The data channel may be comprised of the following components: format conversion devices, encoders (compressors) and decoders (decompressors), buffers, stream segmentation and re-assembly devices, multiplexors and demultiplexors, modulators and demodulators, transmission facilities, switches, multi-point conference units, and other components necessary to achieve the desired channel characteristics.

3.1.13 NTSC Field Integrity: An attribute of a Visual Channel present when the content of odd (even) NTSC Fields in the source sequence is conveyed in the odd (even) fields at the output.

3.1.14 Repeated Video Frame: An output Video Frame that is indistinguishable from its preceding frame(s) in the sequence (when the corresponding input sequence frames possess distinguishable differences). A Repeated Frame is assumed to be generated at some intermediate point in the visual channel. Since Repeated Frames have not traversed the channel from input to output, they are not used in the compilation of the visual channel delay distribution. Repeated Frames also convey no new visual stimulus, and they are excluded from calculation of frame inter-arrival time (and subsequently elementary frame rate).

3.1.15 Non-Repeated Video Frame (Active Frame): An output Video Frame that is distinguishable from its preceding frame(s) in the sequence (when the corresponding input sequence frames possess distinguishable differences). An Active Frame is assumed to have traversed the channel from input to output, and its delay may be included in the visual channel delay distribution. Since Active Frames convey new visual stimulus, they are the basis for calculation of frame inter-arrival time (and subsequently elementary frame rate).

NOTE – Any interpolated frames generated in a decoder will be interpreted as Active Frames in this process.

3.1.16 Frame Matching: The process of comparing one sequence of frames with another sequence of frames in order to determine the correspondence between frames in each sequence and the correspondence of individual frames. See note.

NOTE – One means to test the correspondence between two Video Frames is to compare their digital representations on a pixel by pixel basis, and summarizing over all pixels as the mean-square of the differences (usually called Mean Square Error).

3.1.17 Repeated Video Frame Identification: The process of comparing each output *Video Frame* with its preceding frame(s) in sequence and quantifying the extent of correspondence between each pair. When the correspondence between a pair of frames is high (the only differences are attributable to the noise in the measurement), the pair is indistinguishable; and when the corresponding input sequence of frames possess distinguishable differences, then the current frame is categorized as a *Repeated Frame*. See note, 3.1.16.

3.1.18 Active Video Frame Identification: The process of comparing each output Video Frame with its preceding frame(s) in sequence and quantifying the extent of correspondence between each pair. When there is limited correspondence between a pair of frames (such that the differences measured are distinguishable from the measurement noise), and the corresponding input sequence of frames possess distinguishable differences, then the current frame is categorized as an Active Frame. See note, 3.16.

3.2 Acronyms & Abbreviations

ANSI	American National Standards Institute
ATIS	Alliance for Telecommunications Industry Solutions

ATM	Asynchronous Transfer Mode
EAV	End of Active Video
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GPS	Global Positioning System
ITU-R	International Telecommunication Union, Radiocommunication Sector
LP	Low Pass
MNB	Measuring Normalizing Block
MSE	Mean Square Error
MTIE	Maximum Time Interval Error
NTSC	National Television Systems Committee
ppm	parts per million
PRQC	Network Performance, Reliability, and Quality of Service Committee
RGB	Red, Green, Blue
RMS	Root Mean Squared
SAV	Start of Active Video
SMPTE	Society of Motion Picture and Television Engineers
TIE	Time Interval Error

4 FRAMEWORK OF MEASURABLE PARAMETERS

This clause gives the high-level definitions of the key measurement parameters. There are also method-specific definitions in the clauses that follow.

4.1 Transmission Delay

The time a particular frame takes to traverse the transmission channel. This time is calculated by first recording the times that frames are placed onto the channel, then finding an output frame that has traversed the channel and noting its arrival time at the output. Next, the output frame shall be uniquely matched with an input frame. The transmission delay is then equal to the arrival time minus the input time.

NOTE – When there is little or no activity in the channel, the methods described here will encounter difficulty in making valid measurements. However, this parameter becomes unimportant following the display of the first frame when all succeeding frames are identical (no activity or still video).

4.2 Media Stream Delay Distribution

The set of delays calculated for a sequence of output frames, expressed such that any variation between individual measurements is clearly illustrated. Classical summary statistics may also be supplied, as applicable.

4.3 Active Frame Inter-Arrival Time

The time between successive Active Frames at the output of the channel. This time is calculated by selecting an Active Frame (or Non-Repeated Frame that has traversed the channel) and noting its arrival time at the output. Then the most recent (previous) Active Frame shall be found and its arrival time is noted. The channel Active Frame Inter-arrival Time is then equal to the present frame's arrival time minus the previous arrival time. See note, 4.1.

4.4 Active Frame Inter-Arrival Time Distribution

The set of inter-arrival times calculated for a sequence of active output frames, expressed such that any variation between individual measurements is clearly illustrated. Classical summary statistics may also be supplied, as applicable.

4.5 Elementary Frame Rate

The reciprocal of the Active Frame Inter-arrival Time for the present Active Frame. The Elementary Frame Rate is equal to 1 divided by the difference between the arrival times of the present and previous Active Frames.

4.6 Frame Rate Statistics

A set of statistics that are calculated for a sequence of active output frames, expressed such that any variability is clearly illustrated. When reporting summary statistics for frame rate, they shall be computed using the inter-arrival time distribution, and taking the reciprocal.

5 TEMPORAL CALCULATIONS FOR GENERAL COMMUNICATION MEDIA

This clause gives a general model for calculation of the parameters in this specification and applies the model to each medium covered here. The figure below illustrates the general model. Two sequences of *presentation units*, P and P′, enter the channel input and leave the channel output interfaces. As the last part of each presentation unit passes the interface, the measurement system reads a timer, T, and associates the value T(n) with input presentation unit n, P(n). At the output interface, the measurement system reads a timer, T′, and associates the value T′(m) with output presentation unit m, P′(m).

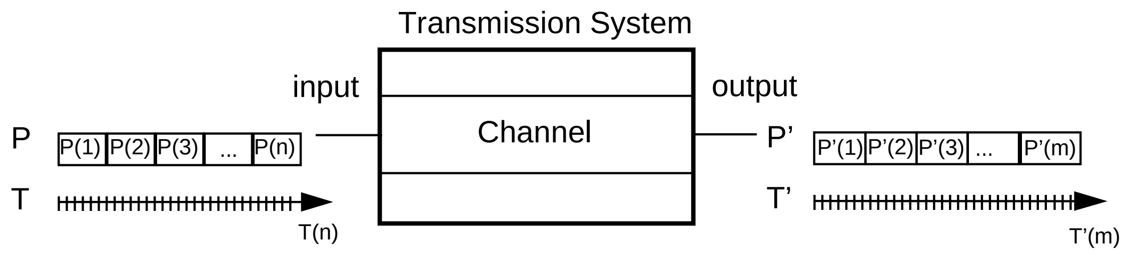

Figure 6 - A General Model for Presentation Units

Note that timers T and T′ can be the same timer in some measurement configurations -- e.g., local and loop back configurations as shown in Figure 3, Figure 4, and Figure 5). Otherwise (e.g., end-to-end configurations as in Figure 2), the timers shall be synchronized within the tolerances given in clause 10. Also, a single timer supplies near-simultaneous time stamps for all channels at a given interface, permitting direct calculations between and within channels.

5.1 Single Channel Calculations

The Channel Delay for presentation unit P′(m), after determining that P′(m) is matched with P(n), is

$$t_P(m) = T'(m) - T(n)$$

Matching is usually trivial in a loss-less channel, but the methods defined here will deal with both distortion and complete loss of presentation units. There shall be differences between the successive presentation units at the input, or matching results will have ambiguity.

The Inter-Arrival Time for presentation unit P′(m)at the channel output is

$$b'_P(m) = T'(m) - T'(m-1)$$

where m-1 is the index of the previous presentation unit.

Note that P and P′ are illustrated as continuous streams above, but non-periodic arrivals are possible with many media. In fact, complete presentation units may take longer to exit the system than enter. This is the key reason for time stamping the last part (see Annex A for further discussion on this topic).

The Elementary Frame Rate for P′(m)at the channel output is

$$f'_P(m) = \frac{1}{T'(m) - T'(m-1)}$$

Complete sets of individual measurements for delay and inter-arrival time, represented as

$$t_P = \{t_P(1), t_P(2), t_P(3), t_P(4), \ldots t_P(M)\}$$

and

$$b_P = \{b_P(2), b_P(3), b_P(4), \ldots b_P(M)\}$$

may be collected for statistical and graphical analysis.

5.2 Media Frames

In the Visual Channel, the presentation units are Video Frames. Clause 6 defines these units for each video interface. V and V′ are the variables for the units of the input and output video streams.

In the Audio Channel, the presentation units are Audio Frames. Audio Frames are a group of digitized samples representing the audio stream (see chart below). Clause 7 defines these frames for audio interfaces. A and A′ are the variables for the units of the input and output audio streams. The recommended audio frame length is approximately the same duration as the associated *Video Frame,* if present -- e.g., 16.66.ms for NTSC).

Table 1 - Preferred Audio Frame Length

samp/sec	preferred sample size, duration
8000	128, 16 ms
16000	256, 16 ms
32000	512, 16 ms
44.1 k	512, 11.61 ms
48 k	512, 10.66 ms

In a Data Channel, the presentation units are Data Frames. Data Units are a unique test word, or a group of bits representing some presentation unit for the user application of the data stream. Clause 9 defines these units for data interfaces. D and D′ are the variables for the units of the input and output data streams.

5.3 Synchronization Calculations

As an example of temporal calculation of synchronization parameters, consider the audio and video media streams where:

- A(m) and V(n) are associated, either by definition of the measurement device or the association is known a priori;
- V′(q) and V(n) are matched, through the methods described in this specification;
- A′(p) and A(m) are matched, through the methods described in this specification.

The time offset between associated audio and Video Frames at the input is

$$o_{AV}(m,n) = T_A(m) - T_V(n)$$

This parameter indicates the position of the audio frame with respect to the Video Frame in time.

The time offset between associated audio and Video Frames at the output is

$$o'_{AV}(p,q) = T'_A(p) - T'_V(q)$$

Note that $o'_{AV}(p,q)$ is the synchronization relationship perceived by a user at the channel output.

The time skew between associated audio and Video Frames at the output, introduced by the transmission system channel, is

$$s_{AV}(p,q) = o'_{AV}(p,q) - o_{AV}(m,n)$$

Following the convention of positive time delay, synchronization time lag is a positive value. If the second channel is leading, the parameter has a negative value.

For systems operating at low frame rates, time skew other than zero may provide perceptually more-accurate lip-sync. For systems with variable video delay, it is usually undesirable to vary the audio delay to maintain zero skew since the audio may become unintelligible through such manipulation.

If a transmission system is capable of multi-channel audio, these calculations are appropriate for assessing synchronization between audio channels.

6 VIDEO MEASUREMENTS

6.1 Collecting Video Frames for Measurements

This clause describes the hierarchy of elements that can exist in a video sequence, and specifies the level of each video coding hierarchy that is the fundamental unit for comparisons. This standard defines these fundamental units as *Video Frames,* re-defining the terms of the coding hierarchy as necessary. We first cover this topic at a conceptual level, to foster extension of the methods beyond the specific electrical interfaces dealt with later.

6.1.1 Description of Video Frames

Many granularity levels of information may be present in a video sequence. The information in video sequences can be divided as shown below:

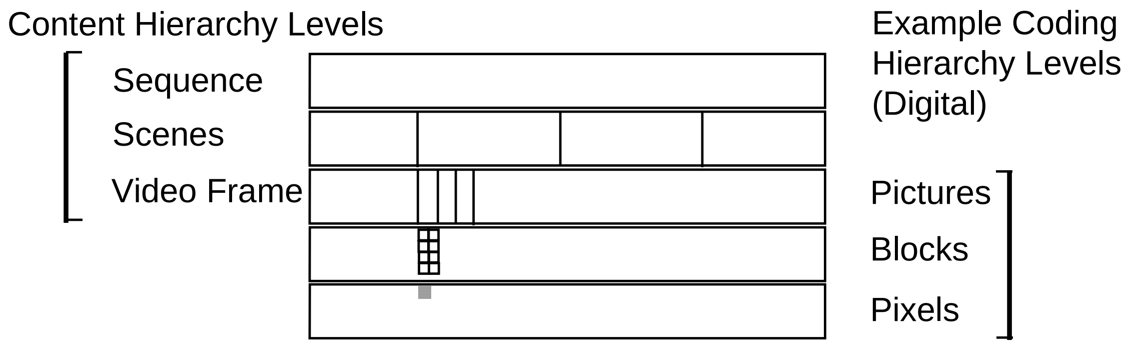

Figure 7 - Example Video Sequence Hierarchy

Figure 7 shows a complete video sequence composed of four scenes with varied content. Each scene is composed of many pictures or frames that differ to convey motion or change. Pictures are the lowest autonomous level of this example Content Hierarchy in the temporal domain, in that the smallest content changes take place at this level over time. This level is the fundamental presentation unit to the user. We define *Video Frames* at this level, where differences between sequential units appear throughout the unit of presentation.

Video Frames may also be at the highest level of the video coding hierarchy. Figure 7 shows an example digital coding hierarchy where this is true. However, if dependent coding between sets of pictures is allowed, then the content and coding hierarchies could overlap by more than one level. Video Frames would continue to be defined at the lowest content level.

6.1.2 Video Frames at the NTSC Composite Interface

Figure 8 shows the video sequence hierarchy for NTSC color signals at the composite interface.

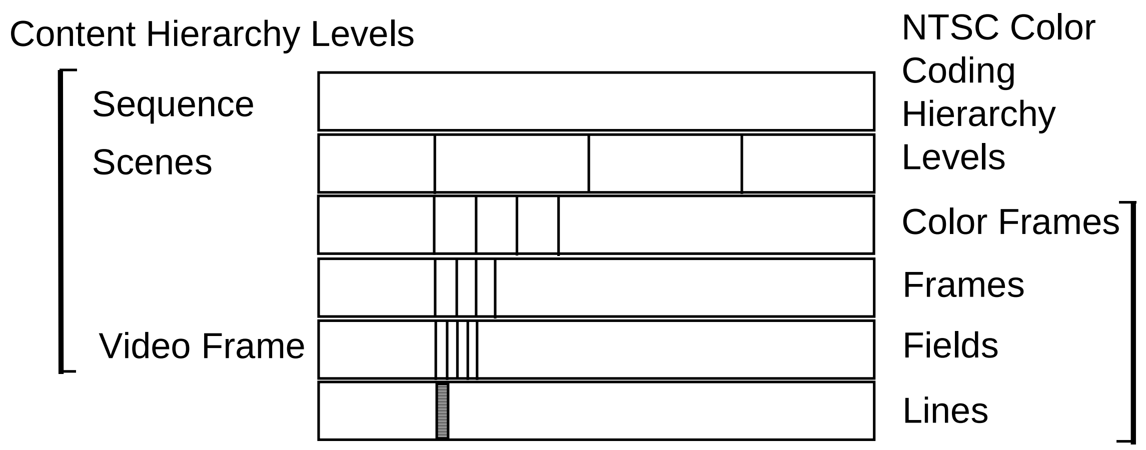

Figure 8 - NTSC Video Sequence Hierarchy

For NTSC interfaces, a *Video Frame* is defined as one NTSC Field, where an NTSC Field is specified in SMPTE 170M. Successive fields may convey new information to the user throughout the unit of presentation, although not on every line of the presentation unit. Also, since the field rate is higher than the frame rate, this definition permits finer sampling of the video sequence.

This definition is also necessary to accommodate transmission systems that (due to system restrictions such as transmission bit rate) convey active Video Frames at rates <30 frames per second and display each new Active Frame as soon as possible. This display update method can result in substantial differences between successive NTSC fields.

6.1.2.1 Analog to Digital Conversion

The methods of measurement described in subsequent subclauses require digitization of the analog NTSC signal. ITU-R Recommendation BT.601-5 provides a high-fidelity method to sample the analog active-line area of the 525-line NTSC luminance signal. Experience has shown that the luminance signal supplies sufficient information to compare and match Video Frames for the measurements in this specification.

Figure 9 shows how the digitized samples corresponding to Video Frames may be organized.

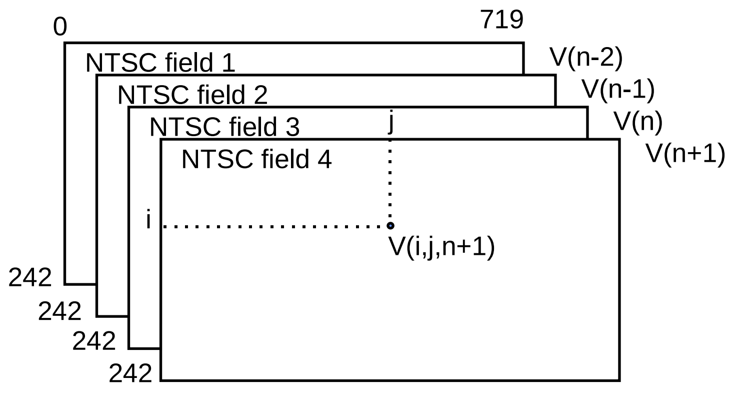

Figure 9 - Recommended Coordinates for Video Frame Digitization

V(n) is Video Frame n at time T(n), V(n-1) is the previous Video Frame, and V(n+1) is the next Video Frame. V(i,j,n) is the (i,j) luminance pixel in Video Frame n at time T(n).

Other coordinate systems are possible, but the numbering from upper left to lower right should be used to facilitate comparisons between different measurement implementations. Implementation of the Digital Component Interface in 6.1.3 will also facilitate these comparisons. As an example of other coordinate systems, SMPTE 170M assigns specific line numbers that begin with vertical synchronization lines.

6.1.2.2 Time Stamp Assignment

The time, T(n), associated with Video Frame n shall be read immediately following the digitization of the last pixel in the frame, coordinate (242, 719), and before the next NTSC line begins (10.222 µs).

6.1.2.3 Gain, Active Area, and Spatial Alignment

This specification requires a correction factor for gain (g), level offset (l), horizontal shift (h) and vertical shift (v) prior to measurement to ensure accuracy.

T1.801.03-2003(R2008) defines a method to measure Average Gain and Level Offset (in 5.1); as well as manual and automated methods to measure Active Video Area (in 5.3) and Active Video Shift (in 5.4).

This specification recommends these methods to ensure the quality of frame to frame comparison and frame matching methods. When measured and used, Active Video Area coordinates and Active Video Shift coordinates shall be specified.

If the transmission system re-sizes the input frames (i.e., expands or contracts the scale in the horizontal, vertical, or both directions), such that additional differences are present when comparing the input and output sequences, then it may be desirable to use a correction factor for the size change (z) to minimize the differences prior to frame matching. When re-sizing is deliberate and extensive, it is essential to compensate for it. It is not known whether significant re-sizing is prevalent in multimedia communication systems, nor at what level the re-sizing contributes significant noise, but there is an opportunity to compensate for re-sizing and continue with the processes as specified here.

Measurement errors may result if these methods are not used. For example, if the channel has a video shift of 4 pixels, a camera pan in the same direction would result in incorrect matching frames with inaccurate delays. If left uncompensated, shifts and gain or level offsets would also increase the noise level and reduce the effectiveness of frame matching based on Mean Square Error.

There are circumstances where simplified methods yield accurate measurements, as shown in the contributions leading to the development of this specification (see Annex C). The following subclause indicates where measurement simplifications are possible.

6.1.2.4 Recognized Options

A recognized option for this specification is to define a sub-frame area in terms of the x-y coordinates and make all comparisons and matching operations on this sub-frame. Alternatively, the sub-frame could be defined by the safe action area and/or safe title area in SMPTE RP 27.3. This allows measurements to exclude frame boundaries without determining the Active Video Area exactly, and improve the capture noise threshold by avoiding error at frame boundaries. If either of these options are adopted, the sub-frame coordinates shall be specified.

When defining a sub-frame area, the following items should be considered:

a) Avoid borders that do not contain picture;

b) Include still areas with both horizontal and vertical edges and a full range of pixel values when possible, to aid with determining spatial correction factors;

c) Include areas with motion, or delay measurements will produce ambiguous results;

d) The size of the sub-frame area in pixels is proportional to computation time. In addition, small areas may improve the frame matching operation when motion is localized, but shall be large enough to avoid ambiguity.

Another option is to use a lower horizontal sampling during digitization for storage. Experience has shown that horizontal resolution of 320 pixels in the digital active-line period can support sufficient quality computation for these measurements, and spatial alignment corrections may not be necessary. If this option is adopted, the horizontal pixels in the digital active-line period shall be specified.

When measuring systems where the picture rate is known to be less than 30 per second, it is permissible to collect every other NTSC Field to reduce capture storage requirements, noting the field selected (even or odd). One example system is ITU-T Recommendation H.320 compatible terminals with video coding according to ITU-T Recommendation H.261. If this option is adopted, the field selected shall be specified.

6.1.3 Video Frames at the 525-line Digital Component Interface

When working with component video, we again define a Video Frame as one field of the 525-line format.

6.1.3.1 Signal Organization

Video signals at the digital interfaces (SMPTE 125M for parallel and SMPTE 259M for serial) have already been converted from their analog form, and no additional digitization is required here. These interfaces multiplex 8 or 10 bit samples of the video components in the order (C_B, Y, C_R, Y, C_B, Y,…). Measurement systems may use only the Y samples.

Timing sequence information words are inserted in the bit stream to identify the digital active line. The active lines begin following the Start of Active Video (SAV) word and end prior to the End of Active Video (EAV) word, and contain 720 luminance samples. The recommended sample coordinates are the same as for the NTSC Composite interface.

6.1.3.2 Time Stamp Assignment

The time, T(n), associated with Video Frame n shall be read immediately following the communication of the last luminance word in the frame, coordinate (242, 719), and before the next line begins (10.222 μs).

6.1.3.3 Gain, Active Area, and Spatial Alignment

The same requirements and options (see 6.1.2.4) for the NTSC Composite Interface apply here as well.

6.2 Mean Square Error Methods of Measurement for Video

This subclause gives the methods of measurement for a system employing a Mean Square Error approach. Implementations of this method shall be able to supply appropriate Video Frame sequences at the channel input. The method also requires capture and, if necessary, digitization of the luminance component of Video Frame sequences at the channel interfaces. Supply and capture shall be conducted in accordance with the provisions of 6.1 for the interfaces in use. Once the Active Frames and their matching frames in the input sequence are found, the desired temporal calculations shall be conducted as specified in clause 5.

The methods accommodate several special circumstances, including high quality transmission systems that maintain interlaced field integrity, and the use of source video sequences derived from the 3:2 pull-down process from 24 fps film.

6.2.1 General

Detecting Active Frames within a sequence of Video Frames, and finding Matching Frames between sequences requires a standard method of comparison. This method compares Video Frames on a pixel-by-pixel basis, and summarizes the difference between a pair of frames as the Mean Square Error over all pixels of interest. Thus, for a pair of frames (one from the input sequence and one from the output sequence) the Mean Square Error (MSE) is

$$M[V'(m), V(n)] = \frac{1}{K_s} \sum_{j=\mathrm{Jmin}}^{\mathrm{Jmax}} \sum_{i=\mathrm{Imin}}^{\mathrm{Imax}} [V'(i,j,m) - V(i,j,n)]^2$$

where V'(i,j,m) is the value of pixel i,j in the output frame at time T'(m), and V(i,j,n) is the value of pixel i,j in the input frame at time T(n). K_s is the total number of pixels in the rectangular sub-frame of interest, given by

$$K_s=(I_{max}-I_{min}+1)\times(J_{max}-J_{min}+1).$$

Note that V'(i,j,m) has been corrected for any gain, level offset, horizontal shift, vertical shift, and spatial scaling (if necessary) between input and output (with corresponding correction factors g, l, h, v, and z):

$$V'(i,j,m) = \frac{V^*(x+v, y+h, m) - l}{g}$$

where V*(x,y,m) is the output pixel before application of the correction factors. If the output video must be re-sized to match the input, then

$$V'(i,j,m) = \frac{V^{**}(\hat{i}+v, \hat{j}+h, m) - l}{g}, \text{ where } V^{**}(m) = f(V^*(m), z)$$

and where $f(V^*(m), z)$ represents a re-sizing function.

For comparisons between adjacent frames within a sequence (e.g., to detect Active Frames at the output interface), V(i,j,n) becomes V'(i,j,m-1) in the equation for MSE, above.

MSE is an important factor in the calculation of Peak Signal-to-Noise Ratio (PSNR).

$$\mathrm{PSNR} = 20\log_{10}\left[\frac{V_{peak}}{\sqrt{M[V'(m), V(n)]}}\right]\mathrm{dB}$$

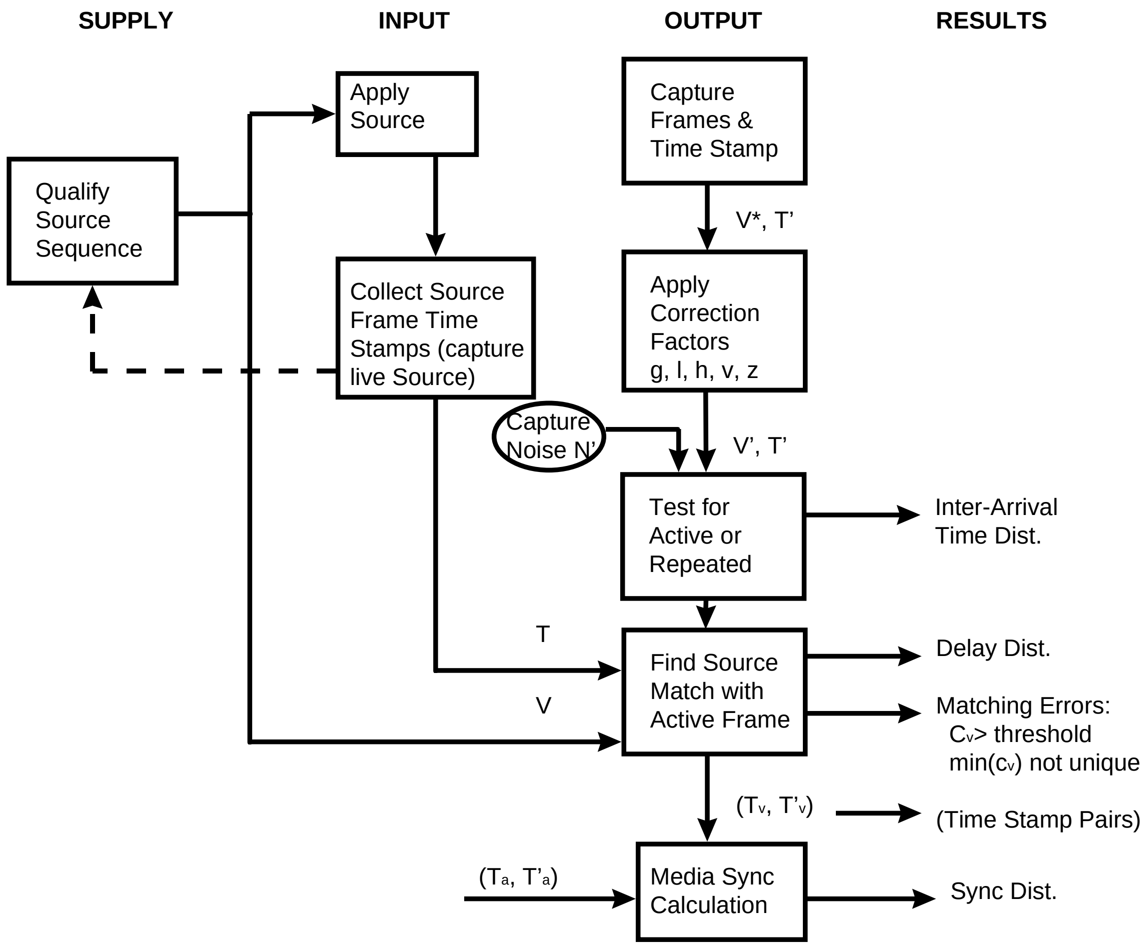
SUPPLY
INPUT
OUTPUT
RESULTS
Qualify Source Sequence
Apply Source
Collect Source Frame Time Stamps (capture live Source)
Capture Frames & Time Stamp
V*, T'
Apply Correction Factors g, l, h, v, z
Capture Noise N'
V', T'
Test for Active or Repeated
Inter-Arrival Time Dist.
T
V
Find Source Match with Active Frame
Delay Dist.
Matching Errors: Cv> threshold min(cv) not unique
(Tv, T'v)
(Time Stamp Pairs)
(Ta, T'a)
Media Sync Calculation
Sync Dist.

Figure 10 - Flow Diagram for MSE-based Video Measurements

Figure 10 illustrates the high-level measurement process for the MSE method.

6.2.2 Calibrating the Minimum Distinguishable Difference Between Frames

This subclause specifies the method to determine the noise (or unwanted variation) in the digitization and storage processes that collect Video Frame sequences for comparison. This noise level is dependent on the specific options chosen (e.g., digitization format) and shall be known in order to make valid measurements.

The test conditions to calibrate the capture noise are as follows:

a) *Apply a **still video** scene to the channel input*. T1.801.02-1996(R2006) (3.1.8) defines still video as "video imagery that conveys no motion or change." It is important to maintain the same input video signal to noise ratio during calibration and measurement. One technique uses a source sequence composed of a single repeated frame from one or more of the motion video sequences intended for further testing. This does not reproduce the noise in the source sequence, and is only appropriate for cases where the capture noise is significantly higher than the source noise. For some test sequences and live video, it may be possible to divide the Video Frame spatially and define a still sub-frame (e.g., background) for this calibration and a motion sub-frame for other measurements. Still test signals (SMPTE Color Bars) have been used successfully as well (again, the source noise present in the still test signal shall be the same as in the testing sequence).

b) *Capture (digitize and store) the corresponding sequence of frames at the channel output*. 30 to 60 frames should be sufficient. When the channel employs digital compression, it shall be allowed to achieve a steady quality level on the still, thereby avoiding any *scene-cut response* (see ANSI T1.801.02, 3.2.14) that would corrupt the noise measurement.

In general, the capture noise at input will be different from the noise at the output. Some codecs filter out source noise to improve the signal for encoding.

For a 30-frame sequence, calculate the set of 30-1=29 adjacent frame MSE values, M[V'(m), V'(m-1)]. The output capture noise level is the maximum MSE value of the set.

$$v'(m) = M[V(m), V'(m-1)] \text{ for } m = 2,3,\ldots 30$$

where v'(m) is the MSE value for frame V'(m), and the capture noise, N', is

$$N' = \max(v')$$

where v' is the set of MSE values for sequence V'. The variation within the set of MSE values should be small (<20%) owing to the averaging over many pixels for each value in the set. For the input sequence, we have N = max(v).

To allow for some margin between the capture noise level and a threshold for detecting Active Frames, we define output frames whose $v'(m)=M[V'(m),V'(m-1)]\leq 1.5\times N'$ to be Repeated Frames. For a source sequence, we define frames whose $v(m)=M[V(n),V(n-1)]\leq 1.5\times N$ to be *Indistinguishable frames*. There may be small differences between repeated or indistinguishable frames, but the measurement system cannot detect them reliably.

The selection and qualification of source sequences for testing shall take this threshold into account. One would not expect to detect Active Frames when frames in the source sequence are indistinguishable to the measurement device. This margin also fosters Active Frame detection with greater confidence.

6.2.3 Testing a Sequence for Distinguishable Differences

For a video sequence V, calculate the set of MSE values v and compare each member of the set with the threshold for indistinguishable frames (1.5×N). All frames V(n) whose $M[V(n),V(n-1)]>1.5\times N$ possess distinguishable differences from their preceding frame. A channel under test shall be supplied with Input frames having distinguishable differences to test for Active Frames and Repeated frames.

When considering source sequences for use with high quality transmission systems that preserve NTSC field integrity, it is more appropriate to compare the current *Video Frame*, V(n), with V(n-2) to pair equivalent NTSC fields and avoid comparison error from spatial offset between fields.

The following procedure gives conditional tests to ensure that a Video Frame possesses distinguishable differences (for source sequences with interlaced fields).

a) Compute M[V(n),V(n-1)];

b) If result is ≤1.5N, declare frames indistinguishable, otherwise continue;

c) Compute M[V(n),V(n-2)];

d) If result is ≤1.5N, declare frames indistinguishable, otherwise continue;

e) Frame V(n) possesses distinguishable differences.

Further considerations on the subject of source or input scene characterization are discussed in 6.2.6. Note that sequences created using a 3:2 pull-down process will fail this qualification, but can still be used under the provisions of 6.2.7.

6.2.4 Categorizing Active Frames and Repeated Frames

For an output video sequence, V′, calculate the set of MSE values M[V′(m),V′(m-1)] and compare each value in the set with the threshold for indistinguishable frames (1.5×N′).

Note that many high quality transmission systems preserve NTSC field integrity, while also introducing minimal distortion. For these systems, it is also appropriate to compare the current *Video Frame,* V′(m), with V′(m-2) to pair equivalent NTSC fields and avoid comparison error from spatial offset between fields. When testing at non-interlaced interfaces or using the recognized options for reduced capture rate and resolution, the comparison with V′(m-2) is probably unnecessary.

A frame V′(m) whose MSE results in M[V′(m),V′(m-1)] and M[V′(m),V′(m-2)]>1.5×N′, in response to an input sequence possessing distinguishable differences, has limited correspondence to either frame and shall be categorized as an *Active Frame.*

A frame V′(m) whose MSE results in M[V′(m),V′(m-1)] or M[V′(m),V′(m-2)]≤1.5×N′, in response to an input sequence possessing distinguishable differences, has high correspondence to V′(m-1) or V′(m-2) and shall be categorized as a *Repeated Frame.*

6.2.5 Testing for Correspondence Between Frames (Matching Frames)

For Active Frame m and an X frame input sequence, calculate the set of X MSE values, M[V′(m),V]. The input frame with the best correspondence is the one that produces the minimum MSE value in the set of:

$$c_V(x) = M[V'(m), V(x)] \quad \text{for } 1 \leq x \leq X$$

c_v is the set of MSE values for frame V′(m) in *comparison* with each frame in sequence V, and the input frame that best matches V′(m) is defined as

$$C_V = \min(c_v)$$

(the minimum error (MSE) represents the maximum correspondence or best match between frames).

A set of rules may improve the matching process and reduce ambiguity. There may be instances where a single Active Frame corresponds closely to more than one input frame. Such cases should be minimized with matching criteria based on pixel comparison methods (MSE), but certain circumstances increase the likelihood of ambiguity. These are:

- ♦ Extreme spatial distortion due to low transmission bit rate, use of a low resolution digital frame format, etc.;
- ♦ Source content – High motion (causing smearing or other distortion), repetitive motion, still intervals within a sequence;
- ♦ Low Active output frame rate leaves many source frames as possible matches;
- ♦ Use of frame interpolation can make matching more difficult.

Experience has shown that an implementation of this method can handle examples of these difficult circumstances.

As an aid to automating this method, the following rules may be helpful to resolve ambiguous matches:

a) *Require one-to-one matching*: No more than one Active Frame can match a given input frame. One possible explanation for a double match is that an Active Frame was falsely detected.

b) *Enforce sequence*: If V'(m) matches V(n), then the next Active Frame V'(m+2) shall match V(n+1), or V(n+2), or V(n+3), etc. V'(m+2) is not permitted to match V(n-1) or V(n), but such an event shall be flagged as a possible error.

c) *Recognize minimum delay*: No matches allowed resulting in delay less than t_{min}. Negative delay is precluded by $t_{min} \geq 0$.

d) *Recognize no-match condition*: Some Active Frames may contain too much distortion to match with the transmitted sequence. Such frames shall be counted and reported, along with the no-match threshold used. Measurement system users shall determine the usual range of matching MSE values for the transmission system under test, and set this threshold above this range.

e) *Diagnostics*: The matching process could be repeated from the opposite end of the sequence to see if fewer ambiguous matches and no-matches occur.

f) *Test next frame*: If the next Active Frame in the output sequence has a unique match in the transmitted sequence, use its match and enforce the rules above on the previous Active Frame.

g) *Select best at random*: When ambiguity still exists following application of the above rules, random selection could be used. However, it is recommended that MSE be calculated with sufficient resolution to minimize such occurrences. Errors introduced in the distribution by a random process should cancel out over a sequence and some summary statistics would be unaffected. Such selections shall be counted and reported.

h) *If the results using a specific scene tend to require extensive intervention and resolution using these rules*, the measurement should be attempted using a different scene.

6.2.6 Source Sequence Qualification for the Mean Square Error Methods

The success of the MSE-based methods depend on the use of appropriate source sequences. As stated above, the source sequence shall possess frame-to-frame differences that are distinguishable to the measurement device, avoid repetitive motion, and avoid intervals of still video that will certainly cause matching ambiguity. When using a sub-frame area, the process shall adopt a similar area as a basis for qualification. The following procedure can determine suitable video clips for measurement:

a) *Take the first Video Frame in the source sequence and compare it* with all other frames in the sequence.
b) *Record* the count and position of all indistinguishable frames (as described in 6.2.2).
c) *Analysis*: The interval between all indistinguishable frames shall be sufficient to resolve input-output alignment ambiguities using prior estimates of frame inter-arrival time and other information. For example, if it is known that transmission delay is <2 sec, indistinguishable frames may appear ≥2 sec apart.
d) *Repeat* the above steps with at least the first X frames (for example, X=60). We place emphasis on the early frames in the sequence because some comparison rule outcomes can be dependent on the prior results.

Also, note that source sequences with similar adjacent frames (distinguishable, but by a small amount) will have greater opportunity to produce ambiguous matches with Active Frames. A test of frame to frame differences will reveal the extent of these similarities, as described in 6.2.3.

T1.801.01-1995 (R2006) specifies a set of source sequences that are appropriate for video teleconferencing/telephony system assessment. However, some sequences contain some still video, and those portions shall be avoided for delay measurement (the still portions are desirable for calibration). Other collections of scenes are available, and may be appropriate for other applications. Even live video is permitted, providing that the measurement system can capture and store both input and output sequences during a measurement, and that the input sequence is subsequently evaluated for suitability. All input sequences shall be evaluated using the specific measurement implementation according to the procedures given here and in 6.2.3.

6.2.7 Considerations for use of 3:2 Pull-Down Source Sequences

There are several conditions that can result in duplicated frames (and hence, indistinguishable frames) in a source sequence:

a) The sequence content is still, conveying no motion or change.
b) A 3:2 pull-down process created the video sequence from film. Every fifth NTSC field is a repetition of the previous corresponding field (even or odd).
c) After a 3:2 pull-down process, the video sequence is edited resulting in non-periodic NTSC field repetitions.

Strictly speaking, the presence of any of these three conditions would cause the source sequence to fail the qualification tests. It is impossible to make either delay or frame rate measurements with a still sequence. However, some measurements are possible with the 3:2 pull-down cases.

There are two problems encountered when applying the MSE method of measurement with 3:2 pull-down sources:

♦ *The process to match Active Frames with their source sequence counterparts is more complex when some source frames are repeated*. If ambiguous matches occur, delay calculations are suspect for that frame. The ambiguity can be resolved by examining the source sequence for repeated frames.
♦ *The actual frame rate may be higher than that calculated using the Active Frames alone*. Some frames in the output sequence categorized as Repeated may have traversed the channel. These frames can be found through a supplementary matching process.

To make measurements with a 3:2 pull-down source sequence, it is necessary to create a record of Repeated Frames in the source sequence, as described in 6.2.4 (but applied to the source sequence). Following the Active/Repeated Frame categorization on the output sequence, we expect that the matching process will pair Active Frames with non-repeated source frames, and in some cases also with Repeated source frames. Rule a) in 6.2.5 requires one-to-one frame matching, so we determine the better match by referring to the record of Repeated source frames and choosing the non-repeated source frame. Since original frames that result in Active Frames always come before their repeats, the matching process shall always follow the normal sequence from beginning to end, rule e) shall never be tried.

Once the matching process has been conducted on all Active Frames, a second matching process with the Repeated frames can begin. This pass considers both the Repeated source frames and any frames unmatched in the first pass, thereby avoiding most of the non-repeated source frames. If a unique match exists between Repeated frames in the source and output sequences (after matching the adjacent Active output frames) then the arrival time of the Repeated frame shall be included in the distribution of output inter-arrival times. Again, one-to-one frame matching shall be enforced, in case a Repeated source frame is duplicated by the transmission system. No temporal calculations will use Repeated output frames without unique matching source frames.

6.2.8 Factors That Influence Measurement Accuracy and Stability

In many video transmission systems, the decoder must supply Video Frames at its output according to a periodic display regimen (such as the NTSC interface). If the input and output display clocks are not synchronized, a buffer must be added in the decoder. When the decoder has a Video Frame ready for display, it must wait until the next output opportunity and thereby increase the overall system delay. We will call this decoder waiting interval the *output delay*.

The output delay is bounded by the interval between display updates. For transmission systems with NTSC interfaces that can update on NTSC field boundaries, the maximum output delay is 16.7 ms. With updates on NTSC frame boundaries, the maximum is 33 ms. The actual output delay will be some random value between 0 and the maximum.

When input and output NTSC clocks have a small frequency offset, the output delay will vary over time. If the frequency offset is constant, the output delay will slew over its range at regular intervals. If the NTSC clocks are synchronized to independent Cesium beam controlled oscillators, the output delay will change <1 ms every 13,900 hours. If the NTSC clock accuracy is derived from independent quartz oscillators (with 2 ppm offset), the output delay will cycle through the entire 33 ms range every 4.58 hours.

Time stamps with sub-NTSC-field resolution permit characterization of the output delay as an inextricable component of the overall transmission system delay.

6.3 In-Frame Time Code Methods of Measurement for Video

In some situations, it may be possible to insert visible symbols in the input video that can be used to uniquely identify each input frame. These symbols will be carried to the system output, and could be used in a simple way to measure the frame rate and delay. These methods, and restrictions on their use, are for further study.

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 clause 5. Figure 11 illustrates the position of Audio Frames within the audio Content and Coding Hierarchies (in this example, coding frames have shorter duration than Audio Frames).

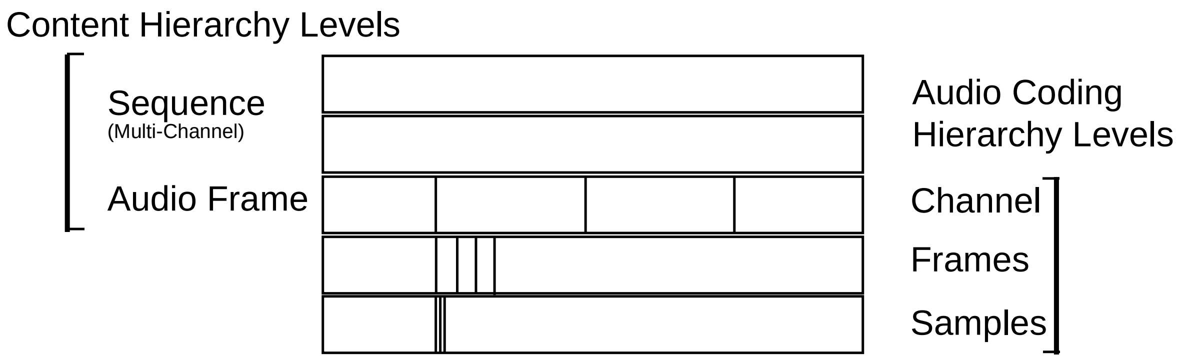

Figure 11 - Audio Content and Coding Hierarchies

7.1.2 Analog to Digital Conversion

The methods of measurement described in subsequent subclauses 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 shall 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 shall 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 shall include appropriate low-pass filtering to prevent aliasing, and shall be matched to 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. The time stamps for each sample within a frame may be computed from the Audio Frame time stamp, since the sample rate is known.

7.2 Delay Measurement for Audio

Many audio channels of potential interest are capable of delivering useable audio signals without preserving audio waveforms. Example audio coding specifications may be found in ITU-T Recommendations G.728, G.729, and G.723.1. This means that a robust delay measurement must not rely on audio waveforms alone. In addition, in packetized audio transmission, end-to-end delay can vary, even over short time scales. Thus a robust delay measurement should be capable of tracking this time-varying delay. A fundamental trade-off permeates time delay measurement: longer delay measurement windows generally provide more robust delay measurements, but longer windows also make it more difficult to respond quickly to changes in delay. Shorter delay windows generally provide less robust measurements, but shorter windows also make it easier to respond quickly to changes in delay. The measurement described here reflects this trade-off.

Figure 12 provides a summary of the core components of the measurement algorithm. A single coarse average delay is first calculated for the entire audio channel output signal vector using smoothed audio 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 measurement where possible. The final output is a vector containing delay measurements spaced at 40 ms intervals across the entire duration of the audio channel output audio signal. Each measurement has full resolution -- i.e., it is to the nearest sample. This section provides more detailed descriptions of each of these stages. The algorithm is sufficiently complex that complete detail in a narrative form is not practical. Thus Annex D provides complete detail in the form of computer code.

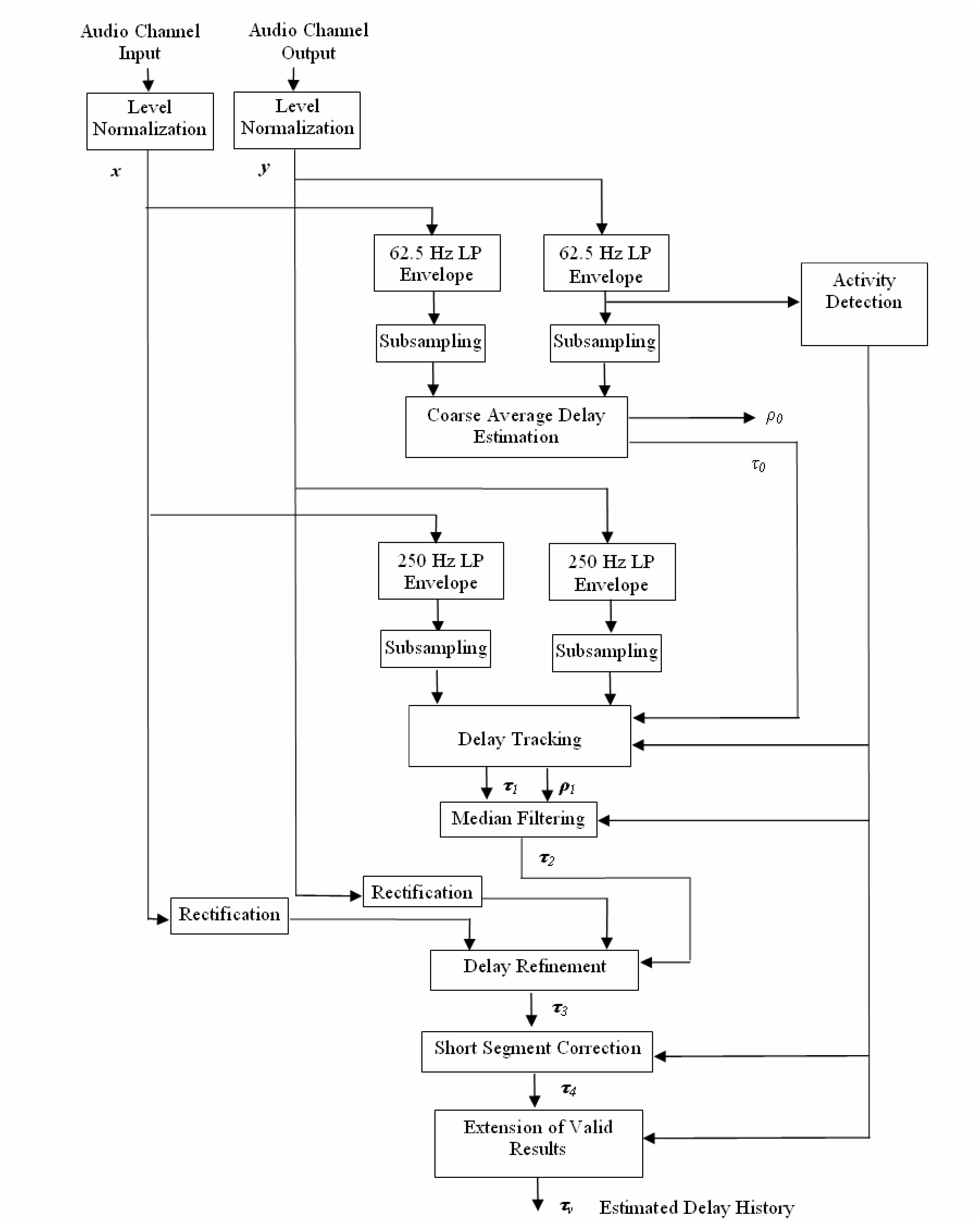

Figure 12 - Functional Block Diagram for Measurement of Time-Varying Audio Delays

7.2.1 Signal Preparation and Level Normalization

A series of Audio Frames must be gathered from the channel input and the channel output before measurement can proceed. 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. The input samples are placed in a vector and the output samples are placed in an identically sized vector. The level normalization stages then normalize these vectors of audio samples to an active audio level of 26 dB below overload. The resulting level-normalized system input and output audio vectors are called $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively.

7.2.2 Coarse Average Delay Measurement

Next, a coarse measurement of average delay is calculated from audio 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 in speech signals, typically leaving a very smooth temporal audio amplitude envelope. The resulting signals are then sub-sampled, resulting in audio envelopes sampled at 125 samples/second. These envelopes are then passed to the coarse average delay measurement stage which applies an FFT-based[4] cross correlation to measure the average delay τ_0 between the two audio envelopes. The corresponding peak correlation value is ρ_0.

7.2.3 Activity Detection

The activity detection stage classifies each sample of $\boldsymbol{y}$ as either active or inactive so that samples in each class can be treated appropriately. To make this classification, the audio 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.

7.2.4 Delay Tracking

The delay tracking stage measures the actual delay of the samples in $\boldsymbol{y}$ as that delay varies about the average delay τ_0. This stage uses audio 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 any speech signals present, thus allowing for its potential use in delay measurement. The resulting signals are then sub-sampled, resulting in audio 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

[4] Background information on Fourier Analysis and applications to speech signal processing can be found in "Digital Processing of Speech Signals," see Annex C.

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 measurement 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 τ_1 of delay measurements spaced at 40-ms intervals in the active portions of the audio signal.

The median filtering stage sweeps a 500-ms window across τ_1. It calculates the median value of all valid delay measurements in the window that have a correlation of at least 0.8. This thresholding helps to minimize the effect that lower quality delay measurements 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 measurements 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 transmission systems. The result of the median filtering stage is a vector τ_2 containing delay measurements spaced at 40-ms intervals. These measurements are available only in active areas where at least some of the correlations exceed 0.8.

7.2.5 Refinement of Delay Measurements

The delay refinement stage seeks to refine the delay measurements available in τ_2 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 τ_2 that contains at least 10 ms of active audio. The use of the absolute value function makes the algorithm completely robust to 180-degree phase changes in the audio signals. The correlation operation searches a range of 9 ms about the delay given in τ_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 τ_3 containing delay measurements spaced at 40-ms intervals. Valid measurements are available consistent with the rules described above.

7.2.6 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:

1. A *pulse* is a segment with measured delay τ_p that has two valid immediate neighbors (left and right) that share a common delay measurement $\tau_n \neq \tau_p$. When a pulse is 280 ms or shorter it is likely that the measured 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 measured delay of τ_n.

2. A *step* is a segment with measured delay τ_s that has two valid immediate neighbors (left and right) with different delay measurements $\tau_l \neq \tau_r$, both of which differ from τ_s. 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 audio in y corresponding to the step segment is correlated with the corresponding rectified audio in 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 measurement 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 measurement of τ_r. If the delay τ_s produces the highest correlation, then the step segment is left unchanged.

3. A *tail* is a segment with a measured delay τ_t that has only one valid immediate neighbor. This neighbor has a delay measurement of $\tau_n \neq \tau_t$. This situation most often happens at the start or end of an active interval. 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 measurement of τ_n.

The result of the short segment correction stage is a vector τ_4 containing delay measurements spaced at 40-ms intervals. Valid measurements are available consistent with the rules described above.

7.2.7 Extension of Valid Measurements

The vector τ_4 contains delay measurements that cover almost all active portions of the signal, with the possible exception of locations where severe localized impairments have corrupted the audio 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 measurement for every single sample in the audio vector $\boldsymbol{y}$, including samples in silent intervals where there is no signal upon which to base a measurement. The final stage of the algorithm extends valid delay measurements to cover segments where no valid measurement has yet been calculated. When such a segment has two neighbors (left and right) with valid delay measurements, then that segment is divided into a *left half* and a *right half*. The valid delay measurement for the left neighboring segment is extended to cover the left half. The valid delay measurement for the right neighboring segment is extended to cover the right half. This stage generates the final output of the variable delay measurement algorithm. This output is a vector τ_v containing delay measurements spaced at 40-ms intervals across the entire duration of the audio channel output signal used in the measurement.

7.2.8 The Special Case of Measuring Fixed Delays

Since longer delay estimation windows provide more robust delay measurements, 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 audio samples in $\boldsymbol{x}$ and $\boldsymbol{y}$, a specialized algorithm can be used. This algorithm performs the level normalization and the average delay estimation described above and then 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 16 ms of τ_0 (the coarse average delay measurement) are considered. If the peak value in $\rho(\tau)$ is 0.73 or greater, that peak location provides the final delay measurement. Otherwise, the system that produced 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 measurement. 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 measurement.

This stage results in a single scalar delay measurement τ_f. The resolution of τ_f depends on the rules described above, and it can be as high as full resolution.

7.2.9 Complete Delay Measurement 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 measurement algorithm (7.2.1 to 7.2.7) or fixed delay measurement algorithm (7.2.8) 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.

A second option would be to apply both algorithms and then apply additional tests to determine which is most reasonable. This option is integrated with the two algorithms described above to create a single complete algorithm. The complete algorithm is summarized in Figure 13. When no prior information is available, the complete algorithm switches appropriately between the fixed and variable delay estimation algorithms. This switching is driven by audio 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 measurements 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 measurement is selected; otherwise the fixed delay measurement 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.

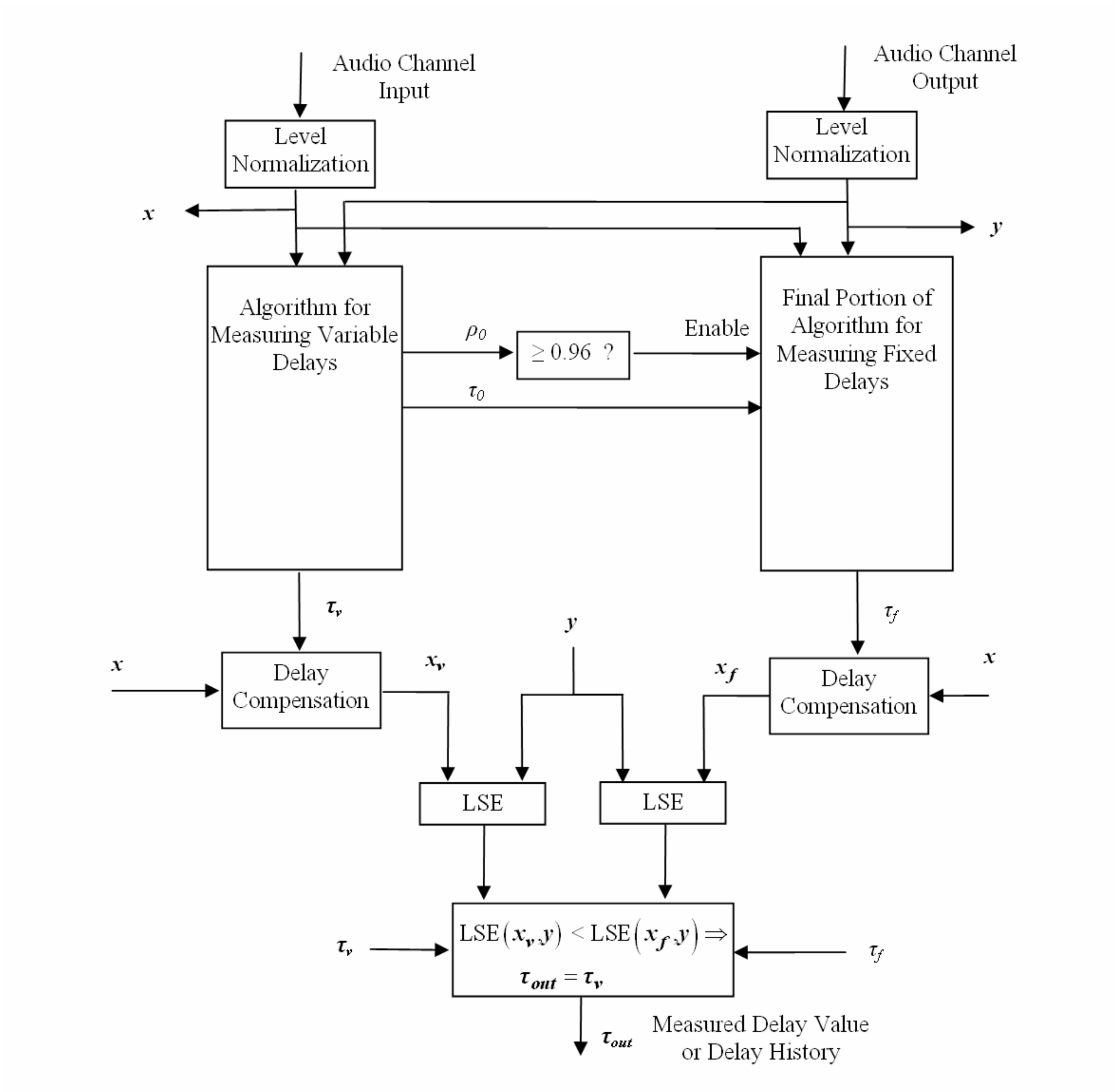

Figure 13 - Functional Block Diagram for Complete Algorithm for Measuring Fixed or Time Varying Audio Delays

7.3 Algorithm Performance

Evaluating the performance of an audio delay measurement algorithm requires care. It is necessary to have true delay histories against which to compare the measurement results. For some channels, at each instant in time, a true delay value can be known with single sample resolution. For other channels, audio waveforms are not preserved. For these channels, at each instant in time, there may be no single true delay value; rather, there is a range of equally true values and that range may cover up to 10 ms. The lack of a single true delay value complicates measurement of algorithm performance. The result is that characterizing delay measurement error in samples or ms has vastly different meanings in different cases.

In light of this, the algorithm described here has been characterized in terms of an average measured quality drop. This characterization was performed for telecommunications bandwidth speech. A sample rate of 8000 samples/sec was used. The speech quality estimation algorithm specified in ANSI T1-518-1998 (R2008) was used. This algorithm, called the *MNB algorithm*, is also described in the technical paper titled "Objective Estimation of Perceived Speech Quality, Part I: Development of the Measuring Normalizing Block Technique" (see Annex C). Channels were simulated by combining numerous different speech codecs with a wide range of circuit-switched and packetized transmission simulations. For each instant of each transmission simulation, a single true value of delay is known. For some of the speech codecs, a single true value of delay is also known. For these codecs, the true transmission delays were combined with the true codec delays to form the true end-to-end delays. For other speech codecs, a range of equally true delay values exists. In these cases, the central value of that range was used, along with the true transmission delay to form a true end-to-end delay.

To characterize the delay measurement algorithm, two speech quality estimates were compared. To generate the first speech quality estimate, channel output speech was delay compensated using the true delay histories and then compared with channel input speech using the MNB algorithm. To generate the second speech quality estimate, channel output speech was delay compensated using the measured delay histories and then compared with channel input speech using the MNB algorithm. Any drop in this second speech quality estimate relative to the first speech quality estimate can be attributed to inaccuracies in the delay measurement process. This characterization involved 12 hours of English language speech from 8 different talkers, 12 different speech codecs, and a wide range of simulated transmission channel conditions. The reduction in estimated speech quality that can be attributed to inaccuracies in the delay measurement algorithm averaged just 0.8% of the full speech quality scale (e.g., 0.03 MOS units on the 5 point MOS scale). This indicates that the delay measurement algorithm provides very close tracking of true delays or delay ranges. This characterization process and some additional development details are described more fully in the technical paper titled "A Bottom-Up Algorithm for Estimating Time-Varying Delays in Coded Speech" (see Annex C).

8 COMBINED AUDIO/VIDEO MEASUREMENT CONSIDERATIONS

Using the concepts and methods defined in clauses 6 and 7, some measurement issues for multiple channels, can be discussed.

8.1 Audio/Video Channel Activity and Synchronization Measurements

Both the video and audio methods of measurement require minimum signal levels in the channels under test in order to produce valid results. Each method has its own specific requirements. Video methods require distinguishable differences between the current and previous Video Frame, while audio methods require the RMS level of a group of Audio Frames to compare favorably with the nominal interface levels.

To be able to compare the audio and video delay measurements, the necessary activity conditions shall be present concurrently at each input, and valid measurements shall be accomplished in both channels (which relies on output activity). Otherwise, synchronization calculations are not possible and a different opportunity shall be sought.

8.2 Associating Individual Measurements

The synchronization calculations of clause 5 require an association between input frames and a frame matching process to yield pairs of time stamps for each channel tested. Calculation of the time offset between channels and the time error introduced by the transmission channel do not compute delay as an intermediate step. Strictly speaking, this prevents comparing delays measured at two different times to assess the synchronization of two channels.

However, audio channels represent a reasonable exception because they carry an isochronous media. When the measured audio delay variation is limited to the expected experimental error, then the average audio delay can be considered a representative constant value. This average delay can be compared with a video or data channel delay distribution to obtain a distribution of time skew between the channels. This allows comparison of audio and video measurements made singly, but under identical source (and other) conditions. This way, test devices that cannot make simultaneous multiple channel measurements may still supply useful information when conditions permit.

If the measured audio delay variation is beyond the expected experimental error, then single channel measurements may not be used.

9 DATA MEASUREMENTS

This clause specifies the methods for delay measurements on data channels that are part of multimedia communications systems. Many data performance standards define delay or transmission time for specific communications protocols (such as ITU-T Recommendation X.25 and the various recommendations defining ATM). This clause references several ANSI T1 standards with pertinent material.

9.1 Collecting Data Frames for Measurement

The Data Channels implemented in multimedia communication systems can vary widely in terms of their purposes and specific attributes. Rather than attempt to deal with all possibilities, this subclause gives the general methods applicable to Data Channels in two main categories (defined below).

9.1.1 Considerations for Defining Data Frames

This standard deals with the Data Channel at a logical level, above the physical layer and its electrical interfaces. However, systems capable of the measurements described in this standard will have test facilities conforming to one or more electrical interface standards. Observations made at these electrical interfaces will be the basis for measurements.

This standard refers to a sequence of Z consecutive bits as a *Data Frame*. Data Frame n is represented by D(n), and the first bit in D(n) is D(1,n). The length of a Data Frame may be determined by the application of the Data Channel.

We consider two possible configurations for user data:

a) Users submit information bits embedded within a standardized structure. These structures may be called packets, cells, or frames;
b) Users submit unstructured streams of bits. The multimedia communication system may perform its own segmentation on this bit stream.

When the user submits a structured bit stream, and that structure permits recognition of individual frames at each channel interface, then the native structure is considered the Data Frame for measurements of multimedia systems. A possible exception is when the native structure contains a large number of bits, and the structure insertion time is large compared with Audio Frames and Video Frames. In this case, it may be more efficient to treat the bit stream as unstructured. The ideal circumstance is equal insertion time for frames in all media, permitting a one-to-one correspondence.

When the Data Channel permits unstructured input bits, and it is possible for the measurement system to supply the bits, then a pseudo-random sequence generator may be used. This gives several advantages:

- The sequence can be generated easily at local and remote sites;
- The repeating length of the sequence can be chosen to avoid ambiguous matches; and
- Data Frame length may be as small as a single multiple of the length of the linear feedback shift register, and may be chosen to closely match the length of other media frames.

When the Data Channel requires a large structure, the pseudo-random sequence generator may supply the information bits carried by the structure.

It may also be possible for the measurement system to collect the input bit stream from the Data Channel's usual source. In this case the Data Frame length will usually coincide with the native structure. When the data source is producing an idle pattern, successful measurements are highly unlikely.

9.1.2 Time Stamp Assignment

The time, $T_D(n)$, associated with Data Frame n shall be read immediately following the communication of the last bit in the frame across the interface and before the next bit is communicated across the interface.

Since frame insertion time is also a useful data transmission measure, additional time stamps may be associated with the first bit of a Data Frame, and shall be read before the next bit is communicated across the interface. Input insertion time may be different from the output insertion time in some systems. Further, insertion time may not be constant.

9.2 Delay Measurement for Data

This subclause gives two methods to match input and output Data Frames.

9.2.1 Matching Structured Data

T1.504-1998 (R2006), (7.3.1.4) discusses methods to determine correspondence among X.25 packets that are applicable to many forms of structured data. Usually, the header bits communicate sufficient information (such as a sequence number), so that packets can be differentiated from one another. These embedded identifiers are a valid basis for matching Data Frames.

If the header information of a specific protocol is insufficient, then it may be possible to match additional identification within the user data field.

9.2.2 Matching Unstructured Bit Streams

T1.517-1995 (R2006) (3.3.5.2) defines correspondence between bits by comparing sequences of bits. Using the symbols of this standard, bits D′(m) correspond to an equal length input sequence if there exist integers n and d such that:

D′(i,m)=D(i+d,n) for almost all integers 1≤i≤Z-d

And:

D′(i,m)=D(i-Z+d,n+1) for almost all integers Z-d+1≤i≤Z

where d is the positive integer offset (d<Z) that may exist between input, D(n), and output, D′(m), frame assignments.

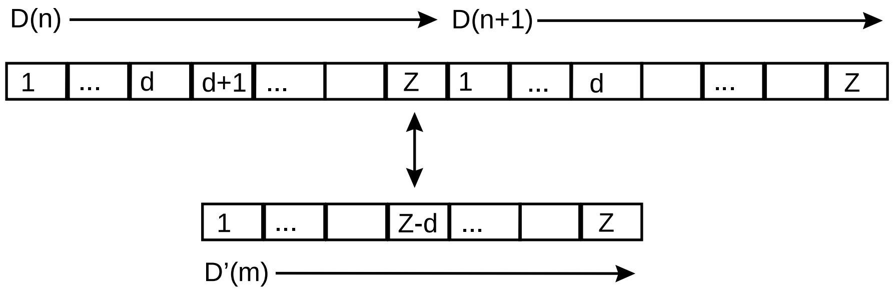

Figure 14 - Correspondence Between Input and Output Data Frames

Figure 14 illustrates correspondence across input frame boundaries.

If the input or output Data Frames are re-aligned such that their bit offset is fixed to d=0, then the correspondence test simplifies to

D′(i,m)=D(i,n) for almost all integers 1≤ i ≤ Z

If bits are communicated across the interfaces in a periodic manner, it is possible to calculate the time stamp for any bit in a Data Frame, making the time stamps available for the first or last bit in any new definition of D(n) that achieves d=0.

Allowing correspondence over *almost all* bits in a Data Frame permits successful matching in the presence of limited bit errors. When there are no bit errors, there shall be equality for all integers i in the range 1 ≤ i ≤ Z.

10 TIMER STABILITY AND SYNCHRONIZATION REQUIREMENTS

This clause gives the minimum specifications for the internal timers or clocks that supply the time stamps for frames. There are two clock configurations to consider:

a) A single clock supplies the input and output time stamps (usually found in local and remote loop-back measurement applications). In this case, only the specifications for accuracy, stability, and resolution apply, since they fully characterize one clock’s performance; or

b) Two clocks, possibly in different (remote) locations at the input and output of the transmission system, supply the time stamps. This configuration applies in the end-to-end measurement application. All specifications of this clause apply in this configuration.

10.1 Resolution

The minimum resolution of the time scale available for inclusion in time stamps is 0.1 μ second (10^{-7} second). This is the intended internal storage resolution for measurements. Although this full resolution shall not be reported when internal clock accuracy does not support it, it allows developers a fine basis for clock stability/accuracy evaluation.

10.2 Accuracy and Stability (Allowable Time Interval Error)

The accuracy and stability of the internal clock time scale is fully constrained with a specification on Maximum Time Interval Error. *Time Interval Error (TIE)* is defined as the time variation of a given time clock's readings with respect to an ideal time scale over a particular observation period, S. *Maximum Time Interval Error (MTIE)* is the largest TIE for all possible measurement intervals within the observation period.

In practice, the transmission measurements conducted according to this standard will last 1 second or more. Therefore, the MTIE specification will begin at 0.01 seconds observation interval (smaller intervals are not specified).

In many applications of this standard, the clock(s) will be synchronized with a time reference signal, such as the *Global Positioning System (GPS)*. In this case, the MTIE is described by the following equation:

$$\text{mtie,ns} \le 10^{-2} S + 150$$

and illustrated in Figure 15 below.

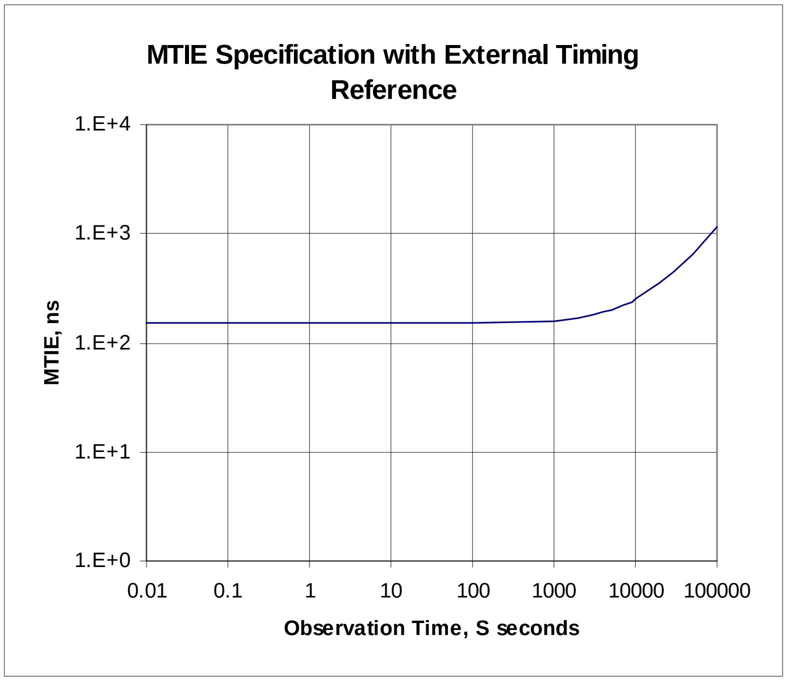

Figure 15 - MTIE with Time Reference

Other applications will use a single internal clock, or a remote clock that can be synchronized before measurement but then relies on its internal -- a.k.a. holdover -- accuracy to maintain time. This permits measurements for some limited time period when a primary timing source -- e.g., GPS -- is unavailable. For Type A internal clocks, the MTIE is constrained by:

$$\text{mtie,ns} \leq 10S + 150$$

and illustrated in Figure 16.

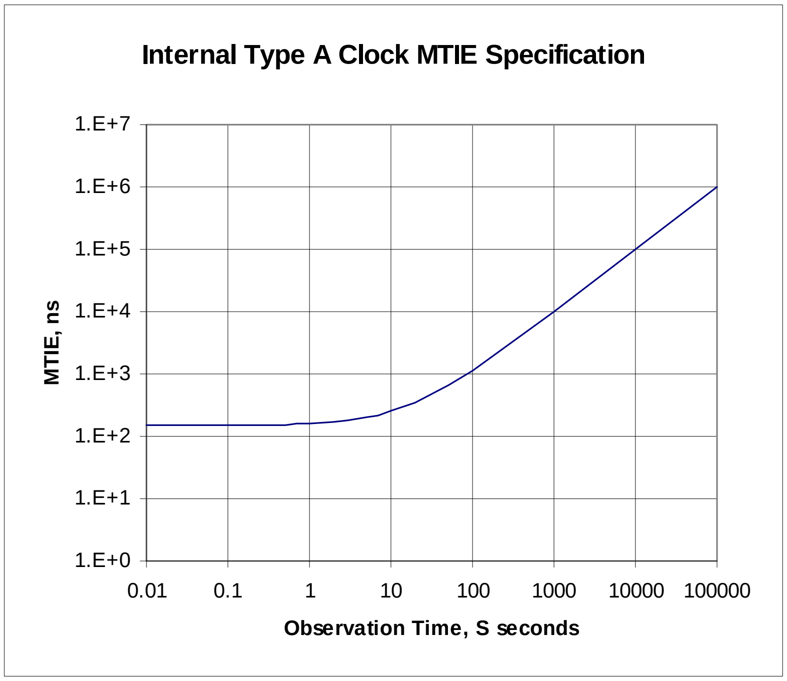

Figure 16 - Internal Type A Clock MTIE Spec. (During Measurements)

For Type B internal clocks, the MTIE is constrained by:

$$\text{mtie,ns} \leq 138.9 \times S + 150$$

and illustrated in Figure 17.

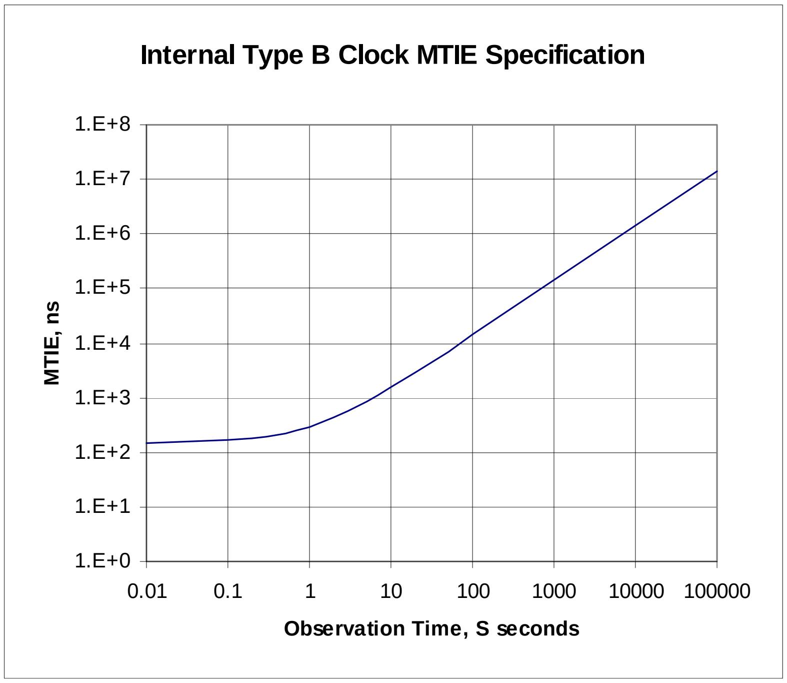

Figure 17 - Internal Type B Clock MTIE Spec. (During Measurements)

In all cases, measurement reports shall be accompanied by the maximum error (determined by the presence of a reference source), including the time elapsed since the reference was available, and the actual measurement interval.

10.3 Time Setting Error

If two or more clocks are used in a measurement, they shall be synchronized. When clocks are synchronized directly, or synchronized to some third reference clock, the maximum setting error will be ±0.075 µsecond (7.5 x 10^{-8} second).

Annex A

(informative)

A TIME STAMP ASSIGNMENT

In clause 5, the specification of time stamps requires their association with the last part of the frame. This annex gives further explanation for this specification, by way of example.

The general cases where time stamp position becomes important are when:

- There are non-periodic frame displays or deliveries (possible with many media); and
- Presentation units take more/less time to exit the system than enter (possible with mixed interface types).

As a practical matter, it is often easier to anticipate the last part of a frame, and therefore, to time stamp the end more easily than the beginning.

It is preferred that our time stamp specification yield delay and synchronization measurements that correspond most closely to the user's experience. Consider a transmission system where:

a) Frame submissions and arrivals are non-periodic;

b) Frames cannot be presented (or displayed) until their arrival at the system output is complete; and

c) Presentation is perceptually instantaneous.

(While these conditions would not hold for all transmission systems, they are possible among systems within the scope of this standard.)

In the graph below, we have frames submitted in a bursty fashion, while the channel tends to smooth the delivery rate. Presentation follows the end of each frame as soon as possible.

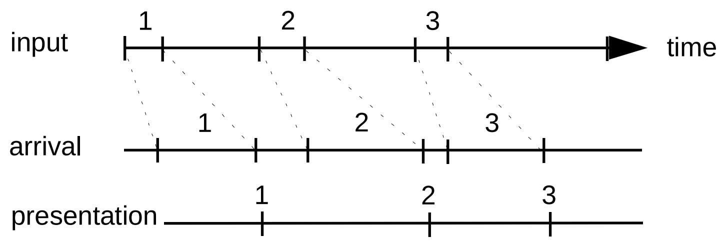

Figure A.1 - Example frame timelines

It is observed that transmission delays for the beginning and end of each frame are considerably different, and that those based on end time stamps are more closely associated with presentation, and also the user-perceived delay.

When display time has a perceptible duration (or when presentation is not instantaneous), the new information must be perceived after the beginning of the frame, but before, or at the end of a frame. In this case, the time stamp for the end of frame bounds the perceived delay.

A simple video example employs a composite PAL input interface (50 fields per second), and an NTSC output interface (60 fields per second). A transmission system with 20 ms delay and a single field buffer is employed, as shown in the graph that follows.

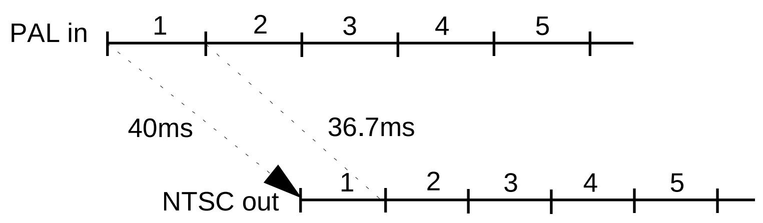

Figure A.2 - Example video format conversion timelines

In this measurement, the delay value (if it is to be corrected for frame insertion time) clearly depends on whether the fields are time stamped at the beginning (40 ms) or end (36.7 ms).

As another example, consider an adaptive system with non-periodic frame delivery controlled by the apparent supply of frames. The system retards frame display when the supply of frames is short, and quickens the pace when frames are plentiful. In this system, any display update variation can be considered part of the delay we wish to characterize, and time stamping the last part of the frame will accomplish this.

Annex B

(informative)

B MATHEMATICAL SYMBOL AND CONVENTION KEY

V	a sequence of adjacent Video Frames at the channel input
V'	a sequence of adjacent Video Frames at the channel output
$V'(m)$	an output Video Frame at time $T'(m)$
$V'(i,j,m)$	luminance pixel (i,j) in output Video Frame m at time $T'(m)$
$V^*(i,j,m)$	output luminance pixel before correction factors are applied
g	gain correction factor
l	level offset correction factor
h	horizontal shift correction factor
v	vertical shift correction factor
z	frame size correction factor
A	a sequence of adjacent audio units at the channel input
A'	a sequence of adjacent audio units at the channel output
D	a sequence of adjacent data units at the channel input
D'	a sequence of adjacent data units at the channel output
T	the 'timer media' stream at the input
T'	the 'timer media' stream at the output
T_P	the set of time stamps associated with general presentation units at the input
T'_P	the set of time stamps associated with output presentation units
$T'_P(m)$	the timer value (time stamp) associated with presentation unit m of a general media stream (at the output)
$T'_P(m-1)$	the timer value of the presentation unit preceding unit m
T_A	the set of time stamps associated with input audio frames
T'_A	the set of time stamps associated with output audio frames
$T'_A(m)$	the timer value (time stamp) associated with output audio frame m
$t_P(m)$	the channel delay for presentation unit m
t_P	a set of channel delays measured for a media stream
$b_P(m)$	the inter-arrival time for presentation unit m
b_P	a set of inter-arrival times measured for a media stream
$f_P(m)$	the elementary frame rate for presentation unit m

M[V′(m), V′(m-1)]	the Mean Square Error (MSE) between two adjacent frames
M[V′(m), V(n)]	the MSE between an output frame and an input frame
$K_s=(I_{max}-I_{min}+1) \times (J_{max}-J_{min}+1)$	the total pixels in the spatial sub-region for MSE
PSNR	Peak Signal to Noise Ratio calculated with peak video level, V_{peak}
v'	the set of MSE values for adjacent output frames within a video sequence
c	the set of MSE values comparing an output frame to the input sequence
c_v	the set of MSE values comparing an output video frame to the input sequence
a'	the set of comparison values for adjacent output frames within an audio stream
d'	the set of comparison values for adjacent output frames within a data stream
N′	is the calibrated output capture noise
C_P	the comparison value for the presentation unit that best matches a specific Active unit
$o_{AV}(m,n)$	time offset between associated audio and Video Frames at the input
$o'_{AV}(p,q)$	time offset between associated audio/Video Frames at the output
$s'_{AV}(m,n)$	time skew between assoc. audio/Video Frames at the output due to the transmission system/channel

B.1 Variables used in Clause 7

LSE($\boldsymbol{x},\boldsymbol{y}$):	log spectral error between the audio signal vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
ρ_0:	value of correlation associated with the delay measurement τ_0
$\boldsymbol{\rho_1}$:	vector of correlation values associated with delay measurements in $\boldsymbol{\tau_1}$
τ_0:	value of the coarse average delay measurement
$\boldsymbol{\tau_1}$:	vector of delay measurements produced by delay tracking stage
$\boldsymbol{\tau_2}$:	vector of delay measurements produced by median filtering stage
$\boldsymbol{\tau_3}$:	vector of delay measurements produced by delay refinement
$\boldsymbol{\tau_4}$:	vector of delay measurements produced by short segment correction stage
τ_f:	final value of fixed delay measurement
$\boldsymbol{\tau_v}$:	final vector of variable delay measurements
$\boldsymbol{\tau_{out}}$:	final delay measurement, selected from τ_f and $\boldsymbol{\tau_v}$
$\boldsymbol{x}$:	vector of normalized samples from audio channel input
$\boldsymbol{x_f}$:	the vector $\boldsymbol{x}$ after delay compensation according to τ_f
$\boldsymbol{x_v}$:	the vector $\boldsymbol{x}$ after delay compensation according to $\boldsymbol{\tau_v}$
y:	vector of normalized samples from audio channel output

B.2 Variables used in Clause 9

Z	length of a Data Frame in bits
D(n)	input Data Frame n
D(i,n)	bit i in input Data Frame n
d	offset in bits between input and output Data Frames when determining correspondence

B.3 Variables used in Clause 10

mtie, ns	Maximum Time Interval Error (MTIE), given in nanoseconds
S	Observation interval for MTIE measurements

Annex C

(informative)

C Bibliography

T1.518-1998 (R2008), *Objective Measurement of Telephone Band Speech Quality using Measuring Normalizing Blocks (MNBs*). [5]

T1.801.01-1995 (R2006), *Digital Transport of Video Teleconferencing/Video Telephony Signals - Video Test Scenes for Subjective and Objective Performance Assessment*.[5]

ITU-R Recommendation BT.500-11, *Methodology for the subjective assessment of the quality of television pictures*.[6]

ITU-T Recommendation H.320, *Narrow band visual telephone systems and terminal equipment.* [7]

ITU-T Recommendation H.261, *Video codec for audiovisual services at p × 64 kbits.* [7]

ITU-T Recommendation P.80, *Telephone transmission quality subjective opinion tests, methods for subjective determination of transmission quality*.[7]

ITU-T Recommendation P.84, *Telephone transmission quality subjective opinion tests, subjective listening test method for evaluating digital circuit multiplication and packetized voice systems*.[7]

ITU-T Recommendation P.861, *Objective quality measurement of telephone-band speech codecs*.[7]

ITU-T Recommendation P.910, *Subjective video quality assessment methods for multimedia applications*.[7]

ITU-T Recommendation P.920, *Interactive test methods for audiovisual communications*.[7]

ITU-T Recommendation P.930, *Principles of a reference impairment system for video*.[7]

A.N. Netravali and B.G. Haskell, *Digital Pictures: Representation and Compression*, Plenum Publishing Corporation, New York, NY, 1988.

L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, Upper Saddle River, NJ, 1978.

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.

S. Voran, *A Bottom-Up Algorithm for Estimating Time-Varying Delays in Coded Speech*, in Proc. 3rd International Conference on Measurement of Speech and Audio Quality in Networks, pp. 43-56, Prague, Czech Republic, 2004.

[5] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS), 1200 G Street N.W., Suite 500, Washington, DC 20005. < https://www.atis.org/docstore/default.aspx >

[6] This document is available from the International Telecommunications Union – Radiocommunication Sector. < http://www.itu.int/ITU-R/ >

[7] This document is available from the International Telecommunications Union – Telecommunication Standardization Sector. < http://www.itu.int/ITU-T/ >

Annex D

(normative)

D AUDIO DELAY MEASUREMENT ALGORITHM

This annex contains a full definition of the audio delay measurement algorithm described in Clause 7. This description is in the form of computer code in the Matlab language. This language provides natural handling of vectors, matrices, and associated operations and thus makes it ideally suited for this signal processing algorithm.

Since this is a high-level language, it resembles pseudo-code to some extent. An important extension is that any variable may be a scalar, a vector, or a matrix, thus allowing for efficient representation of complex signal processing operations. Individual elements of vectors or matrices are accesses as shown in the following examples. The ith value stored in the row or column vector x is extracted to the scalar y by y=x(i). The value stored in the ith row and the jth column of the matrix X is extracted to the scalar y by y=X(i,j). Mathematical operators such as "+," "-," "*," and "/" are extended in the mathematically appropriate ways to apply to scalars, vectors, matrices, or mixtures. The operator ".*" produces element-by-element multiplication, while "./" produces element-by-element division. The symbols "&" and "|" produce the logical operations AND and OR respectively. A single apostrophe indicates the transpose operation. For example, if x is a length n column vector, then x′ is the length n row vector with the same contents. Similarly if X is an m by n matrix, X′ is the n by m matrix formed by exchanging the rows and columns of X. The symbol "…" is used when a line of code is continued on the following line. The symbol "%" precedes each comment.

The main function is ITS_delay_est and this function appears first. This function is supported by 18 other functions, and these appear in alphabetically order in the pages that follow.

```
function Delay_est=ITS_delay_est(x_speech,y_speech,mode)
%Usage: Delay_est=ITS_delay_est(x_speech,y_speech,mode)
%
%This function estimates the delay history for the speech samples in
%y_speech relative to the speech samples in x_speech.
%
%x_speech and y_speech are row or column vectors of speech samples
%with sample rate 8000 samples/sec.  At least 1185 samples (148 ms)
%is required in each vector.
%
%mode='f','v', or 'u' for fixed delay, variable delay or unknown delay type
%
%Delay_est holds estimates of the delay of the speech samples in y_speech
%relative to speech samples in x_speech.  Typically y_speech contains
%samples from the output of some system under test, and x_speech contains
%the corresponding input samples.  Delay_est then holds estimates of the
%delay of that system under test.
%
%Delay_est is 2 by n, with one row for each segment of constant delay.
%Column 1 holds the number of the last sample of a segment of constant
%delay, column 2 holds the estimated delay for that segment. Here are two
%examples:
%
%Example 1 Delay_est=[32000,17] means that a single delay estimate of 17
%samples applies to all 32000 samples in y_speech
%
%Example 2 Delay_est=[12345,-2
%                     32000,400]
%means that for samples 1 through 12345, the delay is estimated to be -2
%samples.  For samples 12346 through 32000, the delay is estimated to be
%400 samples.
%
%In addition, the output [0 0] indicates no delay estimation was possible.
%This can happen when x_speech and y_speech contain unrelated signals,
%or no signal at all.
```

```
%
%Written by Stephen Voran at the Institute for Telecommunication Sciences,
%325 Broadway, Boulder, Colorado, USA, svoran@its.bldrdoc.gov
%May 10,2004



%-----------------------------Speech Input---------------------------------
%Transpose x_speech to form a column vector if necessary
if size(x_speech,2)>1
    x_speech=x_speech';
end

%Transpose y_speech to form a column vector if necessary
if size(y_speech,2)>1
    y_speech=y_speech';
end

%--------------------------Level Normalization-----------------------------
%Measure active speech level
asl_x=active_speech_level(x_speech);
asl_y=active_speech_level(y_speech);
%Force active speech level to -26 dB r.e. overload
x_speech=x_speech*10^(-(asl_x+26)/20);
y_speech=y_speech*10^(-(asl_y+26)/20);



%---------------------Coarse Average Delay Estimation----------------------
[tau_0,rho_0,fir_coeff_63]=coarse_avg_dly_est(x_speech,y_speech);

%Compensate for tau_0, comp_x_speech and comp_y_speech will have same
%length
[comp_x_speech,comp_y_speech]=fxd_delay_comp(x_speech,y_speech,tau_0);

%If compensation for tau_0 results in a speech vector that is shorter
%than 1185 samples, then the input signal vectors x_speech and y_speech
%are not sufficent for delay estimation.  (They may contain unrelated
%signals, no signal, or may simply be too short)
if length(comp_x_speech)<1185
    %Algorithm must terminate
    mode='t';
end

%------------Do further mode determination as necessary/possible-----------
if mode=='u' & rho_0<.96
   mode='v';
end

%-----Fine delay estimation for the fixed and unknown delay cases----------
if mode=='f' | mode=='u'
    %Find fine delay
    fxd_fine_delay=fxd_fine_dly_est(comp_x_speech,comp_y_speech);
    %Find total delay
    D_fxd=tau_0+fxd_fine_delay;
end

%-------Additional stages for the variable and unknown delay cases---------
if mode=='v' | mode=='u'
    %---------------------Speech Activity Detection------------------------
    %Identify active speech samples (active_wf is same size as y_speech,
    %1 indicates activity, 0 otherwise)
    active_wf=find_activity_wf(y_speech,fir_coeff_63);
    %Compensate the activity waveform for tau_0
    [junk,comp_active_wf]=fxd_delay_comp(x_speech,active_wf,tau_0);

    %--------------------------Delay Tracking------------------------------

DCAVS=delay_tracking(comp_x_speech,comp_y_speech,comp_active_wf,150,40,200);

    %--------------------------Median Filtering----------------------------
    SDV=median_filter(DCAVS,500,40,.1,.8);

    %---------------------Combine Results with tau_0-----------------------
    SDV(:,2)=SDV(:,2)+tau_0;     %Add in tau_0 to delay estimates
    if 0<tau_0                   %If tau_0 is a positive quantity
        %then adjust locations of delay segments as well
        SDV(:,1)=SDV(:,1)+tau_0;
    end
    %Adjust final location to exactly match end of y_speech
    SDV(end,1)=length(y_speech);



    %----------------------------Delay Refinement-------------------------
    SDV=delay_refine(SDV,x_speech,y_speech,active_wf,72,.7);

    %---------------Remove Redundant Entries in SDV matrix-----------------
    %If delay and validity do not change from segment n to n+1, then
    %segment n is redundant
    keepers=[find( (diff(SDV(:,2))~=0) | (diff(SDV(:,3))~=0));size(SDV,1)];
    SDV=SDV(keepers,:);
    %Adjust final location to exactly end y_speech
    SDV(end,1)=length(y_speech);

    %-----------------Round Delay Estimates to Nearest Integer-------------
    SDV(:,2)=round(SDV(:,2));

   %-----------------------Short Segment Correction------------------------
    if size(SDV,1)>1
        SDV=short_seg_cor(SDV,x_speech,y_speech,160,280,80);
    end
end



%--------------------------Apply LSE if Necessary--------------------------
if mode=='u'
    [lse_f,lse_v]=LSE(x_speech,y_speech,D_fxd,SDV,16);
    if lse_f <= lse_v
        mode='f';
    else
        mode='v';
    end
end

%---Select Output, Extrapolate Variable Delay Estimate if Necessary--------
if mode=='v'
    Delay_est=extend_val_res(SDV);     %Extrapolate variable delay estimate
elseif mode=='f'
    Delay_est=[length(y_speech) D_fxd];%Reformat fixed delay estimate
else                                    %Mode is 't' for terminate
    %The output [0 0] indicates no delay estimation was possible
    Delay_est=[0 0];
end
```

```
%===========================================================================
function asl=active_speech_level(x)
%Usage:  asl=active_speech_level(x)
%This function measures the active speech levels in the speech vector x.
%
%x is a column vector of speech samples
%asl is the active speech level in dB relative to overload

%Just edit fs to operate at other sample rates
fs=8000;                %samples/second
%code will extend each active region by tau samples (forward in time)
tau=round(.200*fs);
%active speech is defined to be dBth dB below max
dBth=20;

n=length(x);
x=x-mean(x); %mean removal
x=abs(x);
%calculate filter coefficient from time constant
g=exp(-1/(fs*.03));
%perform 2nd order IIR filtering
x=IIRfilter((1-g)^2,[1 -2*g g*g]',x);

at=max(x)*(10^(-dBth/20));      %calculate activity threshold
if at==0
    error('Input vector has no signal')
end

active=x>at;  %find active samples

%Extend each active interval tau samples forward in time
trans=find(abs(diff(active)));
for i=1:length(trans)
    active(trans(i):min(trans(i)+tau,n))=1;
end
```

```
%Test for both activity and non-zeroness to prevent log(0)
x=x( find( 0<x & active ));
asl=20*mean(log10(x))-81;
```

```
%==========================================================================
function [tau_0,rho_0,fir_coeff]=coarse_avg_dly_est(x,y)
%Usage:  [tau_0,rho_0,fir_coeff]=coarse_avg_dly_est(x,y)
%This function generates a coarse delay estimate from speech envelopes.
%This is done in the fs=125 samples/sec domain. (i.e., sub-sampling by 64).
%x and y are column vectors of speech samples
%tau_0 is a delay estimate in samples
%rho_0 is the corresponding correlation value
%fir_coeff is a set of 401 FIR filter coefficients for a 63 Hz LPF

fir_coeff=find_fir_coeffs(400,1/133.33);   %calculate filter coeffs

rx=abs(x);                            %rectify
ry=abs(y);

ex=IIRfilter(fir_coeff',1,abs(x));  %63 Hz LPF, order 400 FIR
ey=IIRfilter(fir_coeff',1,abs(y));
%There is no need to remove filter delay since it is the same in each
%signal

ex=ex(1:64:end);    %Subsample by 64
ey=ey(1:64:end);

%Zero pad so ex and ey have same length
lx=length(ex);
ly=length(ey);
L=max(lx,ly);
ex=[ex;zeros(L-lx,1)];
ey=[ey;zeros(L-ly,1)];

corrlen=length(ex);
m=mean(ex);

%Remove mean of ex from both ex and ey
ex=ex-m;
ey=ey-m;
```

```
%FFT based cross correlation
xc=real(ifft(fft([ex;zeros(corrlen,1)]) ...
    .*fft([flipud(ey);zeros(corrlen,1)])));
[rho,index]=max(xc);

%Convert peak location to a shift
tau_0=64*(corrlen-index);

%Normalize to get cross correlation value
rho_0=rho/((corrlen-1)*std(ex)*std(ey));
```

```
%===========================================================================
function SDVout=delay_refine(SDVin,x_speech,y_speech,active_wf,range,cor_th)
%Usage:  SDVout=delay_refine(SDVin,x_speech,y_speech,active_wf,range,cor_th)
%This function refines the input delay matrix SDVin, to generate SDVout.
%SDVin and SDVout are in the fs=8000 domain and have one row per segment
%of constant delay.
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimation segment (0 for invalid, 1 for valid)
%
%x_speech and y_speech are column vectors of speech samples
%   with sample rate 8000 samples/sec. (x_speech is associated with system
%   under test input, y_speech is associated with system under test output)
%active_wf has the same size as y_speech.  1 indicates speech activity,
%   0 otherwise.
%range gives the search range for this stage in samples
%cor_th is the correlation threshold for refinement

%Rectify speech
x_speech=abs(x_speech);
y_speech=abs(y_speech);

%Copy all data, function will refine it where possible
SDVout=SDVin;
%Find number of segments of constant delay
nsegs=size(SDVin,1);

%Loop over all segments
for i=1:nsegs
    %Find first sample of current segment (first segment is special case)
    if i==1
        start=1;
    else
        start=SDVin(i-1,1)+1;
    end
    %Extract last sample of current segment
```

```
stop=SDVin(i,1);
%Extract delay of current segment
delay=SDVin(i,2);

%Attempt refinement only if there is at least 10 ms of active speech
%in the segment and segment has a valid delay estimate
if 80 <= sum(active_wf(start:stop)) &  SDVin(i,3)==1

    %If segment length is at least 200 ms, use fft-based correlation
    if 1600 <= (stop-start+1)
        %Find delay compensated starting sample in x_speech
        sstart=start-delay;
        %If it is before the start of x_speech, it will be necessary to
        %modify the starting place in both x_speech and y_speech
        if sstart<1
            %Number of samples involved in the modifications
            trim=1-sstart;
            %First sample of x_speech for this segement
            sstart=sstart+trim;
            %First sample of y_speech for this segement
            start=start+trim;
        end

        %Find delay compensated ending sample in x_speech
        sstop=min(stop-delay,length(x_speech));

        %If there is at least 10 ms in both the x_speech segment and
        %the y_speech segment after these adjustments
        if (80 <= sstop-sstart+1) &  (80 <= stop-start+1)
            %Perform FFT-based cross correlation on appropriate
            %portions of x_speech and y_speech
            [un_corr,denom]=fft_xc(x_speech(sstart:sstop), ...
                                   y_speech(start:stop),-range,range);
            %Locate peak in unnormalized correlation
            [peak,loc]=max(un_corr);
            %If correlation value meets or exceeds threshold, or the
            %segment is longer than 1 second
            if (cor_th <= peak/denom) | (8000 < stop-start+1)
                %Apply the refinement
                SDVout(i,2)=delay+(loc-range-1);
            end
        end
    %Segment is less than 200 ms long, use direct-form correlation
    else
        %Find starting sample in x_speech, compensated for
        %delay and search range
        sstart=start-delay-range;
        %If it is before the start of x_speech, it will be necessary to
        %modify the starting place in both x_speech and y_speech
        if sstart<1
            %Number of samples involved in the modifications
            trim=1-sstart;
            %First sample of x_speech for this segement
            sstart=sstart+trim;
            %First sample of y_speech for this segement
            start=start+trim;
        end

        %Find ending sample in x_speech, compensated for
        %delay and search range
        sstop=stop-delay+range;
        %If it is beyond the end of x_speech, it will be necessary to
        %modify the ending place in both x_speech and y_speech
        if length(x_speech)<sstop
            %Number of samples involved in the modifications
            trim=sstop-length(x_speech);
            %Last sample of x_speech for this segement
            sstop=sstop-trim;
            %Last sample of y_speech for this segement
            stop=stop-trim;
        end
```

```
            %If there is at least 10 ms in the y_speech segment after
            %these adjustments
            if 80 < stop-start+1

                %Perform direct-form cross correlation on appropriate
                %portions of x_speech and y_speech
                [un_corr,denom]=non_fft_xc_all(x_speech(sstart:sstop), ...
                                                     y_speech(start:stop));
                %Locate peak in unnormalized correlation
                [peak,loc]=max(un_corr);
                %If correlation value meets or exceeds threshold
                if cor_th <= peak/denom
                    %Apply the refinement
                    SDVout(i,2)=delay+(range-loc+1);
                end
            end
        end
    end
end
```

```
%===========================================================================
function DCAVS=delay_tracking(x,y,active_wf,winlen,winstep,range);
%Usage: DCAVS=delay_tracking(x,y,active_wf,winlen,winstep,range);
%This function does delay tracking (delay in speech signal y relative to x)
%winlen is the length of window used for the delay estimation alg (in ms)
%winstep is the step size between windows   (in ms)
%range is the half-width of the search range (in ms);
%DCAVS is a nwins by 5 matrix, where nwins is the number of delay
%estimation windows that can be fit into y
%Column 1 holds Delay values (in the 16:1 subsampled domain)
%Column 2 holds the corresponding Correlation values
%Column 3 holds the corresponding Activity levels values
%Column 4 holds the corresponding Validities (1 for valid delay estimate,
%   0 otherwise)
%Column 5 holds the Sample number in y (in the 16:1 subsampled domain)
%that is at the center of each window

%Create coefficients for an FIR LPF with 129 taps and a delay of
%64 samples.  Response is -51 dB at 250 Hz.
fir_coeffs=find_fir_coeffs(128,1/32);

%Rectify and filter
x=IIRfilter(fir_coeffs',1,[abs(x);zeros(64,1)]);
%Remove filter delay and subsample by 16
x=x(65:16:end);
nx=length(x);

%Rectify and filter
y=IIRfilter(fir_coeffs',1,[abs(y);zeros(64,1)]);
%Remove filter delay and subsample by 16
y=y(65:16:end);

%Subsample activity waveform
active_wf=active_wf(1:16:end);

%Force subsampled activity waveform to have same length as subsampled y
```

```
ny=length(y);
len_diff=length(active_wf)-ny;
if 0<len_diff
    active_wf=active_wf(1:ny);      %Trim final samples
elseif len_diff<0
    active_wf=[active_wf;zeros(len_diff,1)];    %Zero pad
end

%Convert parameters from ms to samples (in the subsampled domain)
winlen=round(winlen/2);
winstep=round(winstep/2);
range=round(range/2);

%Find total number of window positions that can be placed on y
nwins= floor((ny-winlen)/winstep)+1;

DCAVS=zeros(nwins,5);

%Find locations of centers of windows (sample number in subsampled domain)
first=(winlen+1)/2;
last=first+winstep*(nwins-1);
DCAVS(:,5)=[first:winstep:last]';

for i=1:nwins                                          %Loop over all windows

    start=1+(i-1)*winstep;      %First sample in window
    stop=start+winlen-1;        %Last sample in window
    %Find activity level in window
    DCAVS(i,3)=sum(active_wf(start:stop))/winlen;

    %Assume delay estimation is doable unless one of tests that
    %follows fails
    doable=1;
    if start-range < 1   |   min(nx,ny) < stop+range
       %Not enough samples to do delay estimation
        doable=0;
    elseif std(y(start:stop))==0 | std(x(start-range:stop+range))==0
        %No variation in samples
        doable=0;
    end


    if doable
        %Perform direct-form cross correlation
        [xc,denom]=non_fft_xc(x(start-range:stop+range), ...
            y(start-range:stop+range),-range,range);
        %Identify peak
        [maxrho,index]=max(xc);
        %Calculate corresponding delay in samples
        DCAVS(i,1)=(index-range-1);
        %Calculate corresponding correlation value
        DCAVS(i,2)=maxrho/denom;
        %Mark that window as having a valid delay estimate
        DCAVS(i,4)=1;
    end

end
```

```
%===========================================================================
function SDVout=extend_val_res(SDVin);
%Usage:  SDVout=extend_val_res(SDVin)
%This function extrapolates valid results to cover areas where there are
%none.  SDVin and SDVout are Delay history matrices in the fs=8000 domain:
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimation segment
%           (0 for invalid, 1 for valid)
%
%For interior invalid regions, the function splits the region in half and
%extrapolates each neighboring valid region to cover half of the invalid
%region.  For exterior invalid regions, the function extrapolates the
%single neighboring valid region

%Find number of segments
nsegs=size(SDVin,1);
%Copy input data to output data
SDVout=SDVin;

%If there is more than one segment, then loop over all segments
if 1< nsegs
    for i=1:nsegs
        %If the current segment is not valid
        if SDVout(i,3)==0
            %Leading invalid segment case
            if i==1
                SDVout(1,2)=SDVout(2,2);
            %Trailing invalid segment case
            elseif i==nsegs
                SDVout(nsegs,2)=SDVout(nsegs-1,2);
            %Interior invalid segment case
            else
                %Half the width of the invalid segment
                hw=round((SDVout(i,1)-SDVout(i-1,1))/2);
                %Extend previous segment to cover first
                SDVout(i-1,1)=SDVout(i-1,1)+hw;
                %Copy delay of following segment to cover second half
                SDVout(i,2)=SDVout(i+1,2);
            end

        end
    end

    %Find segments associated with changes in delay
    keepers=[(find(diff(SDVout(:,2))~=0));nsegs];
    %Retain only those segments
    SDVout=SDVout(keepers,[1 2]);
else
    SDVout=SDVout(1,[1 2]);
end
```

```
%===========================================================================
function [un_corr,denom]=fft_xc(x,y,min_d,max_d)
%Usage: [un_corr,denom]=fft_xc(x,y,min_d,max_d)
%This function does an fft-based cross correlation on the waveforms in the
%column vectors x and y.  These two vectors need not have the same length,
%but the resulting delay estimated is defined relative to zero time offset
%of x(1) and y(1).
%
%If possible, an unnormalized correlation value is returned for all delay
%values from min_d to max_d, inclusive.  (If this is not possible, it is
%an error condition)
%
%Thus un_corr is a length max_d-min_d+1 column vector.  When it is divided
%by denom (typically after finding a max), correlation values result.
%un_corr(1) corresponds to min_d, un_corr(end) corresponds to max_d
%The legal search window is approximately +/- min(length(x),length(y).

%Pad with zeros so x and y have same length
lx=length(x);
ly=length(y);
L=max(lx,ly);
x=[x;zeros(L-lx,1)];
y=[y;zeros(L-ly,1)];

corrlen=length(x);

%Remove the mean of x from each signal
m=mean(x);
x=x-m;
y=y-m;

%Perform FFT-based cross correlation
xc=real(ifft(fft([x;zeros(corrlen,1)]).*fft([flipud(y);zeros(corrlen,1)])));

xc=flipud(xc);      %reverse the column vector, top to bottom
```

```
%Test to see if requested values of delay are available
if corrlen+min_d+1<1 | 2*corrlen < corrlen+max_d+1
   error('Not enough input samples to calculate requested delay values.')
end

%Extract requested values of delay
un_corr=xc(corrlen+min_d+1:corrlen+max_d+1);

%Calculate the denominator
denom=(corrlen-1)*std(x)*std(y);
```

```
%==========================================================================
function active_x=find_activity_wf(x,fir_coeff)
%Usage:  active_x=find_activity_wf(x,fir_coeff)
%This function generates an output column vector (active_x) that shows the
%speech activity waveform of the input column vector (x).
%Speech activity is defined via a smoothed speech envelope with nominal
%bandwidth of 63 Hz.  The required FIR filter coefficients are given in
%fir_coef. Samples of the envelope that are within 35 dB of
%the envelope peak are associated with active speech.  Each interval of
%activity is extended by tau samples forward and backward in time.

th=35;   %Threshold for activity detection in dB below peak
tau=800; %Number of samples to extend each active segment in each direction
nx=length(x);

%FIR filter rectified speech
x=IIRfilter(fir_coeff',1,[abs(x);zeros(200,1)]);
x=x(201:end);                                  %Remove filter delay

%Find samples that are above threshold,1 means active, 0 means not active
active_x = (x >= 10^(th/20));

%Extend all active regions by tau samples in each direction
trans=find(abs(diff(active_x)));    %List of transition points
for j=1:length(trans)               %Loop over all transitions
   %Extend activity in each direction
   active_x(max(trans(j)-tau,1):min(trans(j)+tau,nx))=1;
end
```

```
%==========================================================================
function b=find_fir_coeffs(order,cutoff)
%Usage: b=find_fir_coeffs(order,cutoff)
%This function generates filter coefficients for an FIR LPF with
%the specified order and cutoff frequency. Cutoff frequency
%is specified relative to Nyquist frequency. b is a row vector
%of length order+1 that holds the filter coefficients

%Create column vector that holds Hamming window of length order+1
n=order+1;
h=0.54-0.46*cos(2*pi*[0:n-1]/(n-1));

%Create column vector of time-domain indices
t=[-order*cutoff/2:cutoff:order*cutoff/2];

%Calculate sin(pi*x)/(pi*x) (sinc function) for time domain indices
%sin(pi*0)/(pi*0) is defined to be 1
good=find(t~=0);
sinxox=ones(1,order+1);
sinxox(good)=sin(pi*t(good))./(pi*t(good));

%Filter coefficients are product of window and sinc function
b=h.*sinxox;

%Normalize coefficients for unity gain in passband
b=b/sum(b);
```

```
%===========================================================================
function [ptr,seg_type]=find_smallest_seg(SDVLS)
%This function finds the smallest segment that has status 0 and reports
%the segment type.  SDVLS is a matrix with one row per segment of constant
%delay.  The 5 columns are:
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimation segment (0=invalid, 1=valid)
%Column 4, Length of segment in samples
%Column 5, Status of segment (0=needs to be considered, 1=should be
%   ignored)
%
%ptr points to the row in SDVLS that has the smallest segment with
%status 0.  If all segments have status 0 , then ptr=0.  If there is a tie
%for shortest segment, the first segment to occur is reported
%
%seg_type indicates the type of segment ptr points to:
%'IV' segment with invalid delay estimate
%'IS' isolated, valid but neighbors on each side are invalid
%'LT' lefthand tail, the segment to the left of an LT segment is either
%   invalid or does not exist, but segment to right is valid
%'RT' righthand tail, the segment to the right of an RT segment is either
%   invalid or does not exist, but segment to left is valid
%'BI' blip, segment and both neighbors are valid.  Both neighbors share a
%   common delay value which is different from the delay value of the BI
%   segment
%'SP' step, segment and both neighbors are valid and all 3 have different
%   delay values
%'xx' is returned when ptr=0, i.e. there is no result to report.

%Initialize ptr
ptr=0;
%Find number of segments
nsegs=(size(SDVLS,1));

%Create list of all segments with status=0
```

```
goodlist=find(SDVLS(:,5)==0);
%If no such segment, function is done
if isempty(goodlist)
    ptr=0;
    seg_type='xx';
%There is one or more segments with status=0
else
    %Find the shortest such segment
    [dud,loc]=min(SDVLS(goodlist,4));
    %Set ptr accordingly
    ptr=goodlist(loc);
    %If segment is invalid, mark it as such and function is finished
    if SDVLS(ptr,3)==0
        seg_type='IV';
    %Special case for first segment
    elseif ptr==1
        %If segment to right is valid
        if  SDVLS(ptr+1,3)==1
            %First segment is a left tail
            seg_type='LT';
        else
            %Otherwise first segment is isolated
            seg_type='IS';
        end
    %Special case for last segment
    elseif ptr==nsegs
        %If segment to left is valid
        if SDVLS(ptr-1,3)==1
            %Last segment is a right tail
            seg_type='RT';
        else
            %Otherwise last segment is isolated
            seg_type='IS';
        end
    %All remaining segments have two neighbors
    else
```

```
        %Check validity of segment to left of current segment
        lv=SDVLS(ptr-1,3);
        %Check validity of segment to right of current segment
        rv=SDVLS(ptr+1,3);
        %Use these two validities to identify appropriate segment type
        if lv==1 & rv==0
            seg_type='RT';
        elseif lv==0 & rv==1
            seg_type='LT';
        elseif lv==0 & rv==0
            seg_type='IS';
        %Both neighbors are valid, so current segment is either a blip
        %or a step
        else
            %If neighbors have the same delay, current segment is a blip
            if SDVLS(ptr-1,2)==SDVLS(ptr+1,2)
                seg_type='BI';
            %Otherwise current segment is a step
            else
                seg_type='SP';
            end
        end

    end
end
```

```
%==========================================================================
function [source,distorted]=fxd_delay_comp(source,distorted,delay)
%Usage: [source,distorted]=delay_comp(source,distorted,delay)
%This function compensates the source-distorted signal pair by the given
%delay value in samples. The returned source and distorted signals will
%have the same length.

sstart=max(1,1-delay); %source starting point
dstart=max(1,1+delay); %distorted starting point

%Number of samples available
samples=min(length(source)-sstart+1,length(distorted)-dstart+1);

%Extract proper portions
source=source(sstart:sstart+samples-1);
distorted=distorted(dstart:dstart+samples-1);
```

```
%===========================================================================
function D=fxd_fine_dly_est(x,y);
%Usage:  D=fxd_fine_dly_est(x,y);
%This function performs an FFT-based cross correlation on the rectified
%speech signals and then processes the results to find a delay estimate
%x and y are column vectors of speech samples
%D is the estimated delay

range=128;  %half width of range of samples to analyze in this stage

%number of samples to feed into filter while waiting for it to stabilize
headlen=500;
%number of samples (a tail) long enough to cover the longest filter delay
taillen=200;

%find cross correlation of rectified speech
[xc,denom]=fft_xc(abs(x),abs(y),-(range+headlen),range+taillen);

%extract relevant portion
txc=xc(headlen+1:headlen+1+2*range);

%find max
[maxrho,index]=max(txc);
maxrho=maxrho/denom;

if .73<maxrho                %For high correlations, no smoothing is required
    D=index-1-range;         %Calculate delay estimate
elseif .67<maxrho            %For medium correlations, some smoothing helps
    m=64;
    flen=3*m;
    fir_coeff=find_fir_coeffs(flen,1/m);      %Filter lengths are even
    sxc=IIRfilter(fir_coeff',1,xc);
    %smoothed version of cross-correlation function with filter delay
    %removed
    sxc=sxc(headlen+(flen/2)+1:headlen+(flen/2)+1+2*range);
    [dud,index]=max(sxc);
    D=index-range-1;        %Calculate delay estimate
else                       %For lower correlations, more smoothing helps
    m=128;
    flen=3*m;
    fir_coeff=find_fir_coeffs(flen,1/m);      % filter lengths are even
    sxc=IIRfilter(fir_coeff',1,xc);    %More Transparent
    %smoothed version of cross-correlation function with filter delay
    %removed
    sxc=sxc(headlen+(flen/2)+1:headlen+(flen/2)+1+2*range);
    [dud,index]=max(sxc);
    D=index-range-1;        %Calculate delay estimate
end
```

```
%===========================================================================
function y=IIRfilter(b,a,x)
%Usage: y=IIRfilter(b,a,x)
%This function implements an IIR filter in direct form:
%a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
%x and y are column vectors and have the same length
%a and b are column vectors of filter coefficients as defined above
%For FIR filtering, set a=1.
%Note that use of the built-in Matlab function “filter” will result
%in much faster execution

%Normalize b coefficients and reverse their order top to bottom
b=flipud(b/a(1));

%Normalize a coefficients, remove a(1) and reverse the order of
%the remaining coefficients, top to bottom
a=flipud(a(2:end)/a(1));

%Check vector lengths
na=length(a);
nb=length(b);
n=length(x);

%If no "a" coefficients remain, this is the FIR case
if na==0
    %Initialize x and y
    x=[zeros(nb-1,1);x];
    y=zeros(n,1);
    %Loop over all samples in y
    for i=1:n
        y(i)=x(i:i+nb-1)'*b;
    end
%If "a" coefficients remain, this is the IIR case
else
    %Initialize x and y
    m=max(na,nb);
    x=[zeros(m,1);x];
    y=zeros(n+m,1);
    %Loop over all relevant samples in y
    for i=m+1:m+n
        y(i)=x(i-nb+1:i)'*b - y(i-na:i-1)'*a;
    end
    %Extract relevant portion of y
    y=y(m+1:end);
end
```

```
%===========================================================================
function [lse_f,lse_v]=LSE(s,d,Df,Dv,maxsp);
%Usage: [lse_f,lse_v]=LSE(s,d,Df,Dv,maxsp)
%This function calculates log-spectra error for fixed and variable delay
%estimates.
%
%s is a column vector of source speech samples (system under test input)
%d is a column vector of distorted speech samples (system under test
%output)  s and d should have no delay compensation applied
%Df is a fixed scalar delay value
%Dv is a delay history matrix where each row corresponds to a segment of
%constant delay and
%Column 1 holds number of last sample in segment
%Column 2 holds delay value for segment
%Column 3 holds 1 to indicate valid delay estimate for the segment,
%   and holds 0 otherwise
%max sp tells the max spacing between LSE computation locations in ms
%lse_f and lse_v are the fixed and variable LSE results in dB
%
%If the lengths of s and d are such that delay compensation by either
%Df or Dv leaves insufficent signal for LSE calculations, then this
%function returns lse_f=lse_v=0.

%Length of LSE window in samples
lsewin=128;

%Convert from ms to samples
maxsp=round(maxsp*8);

%Find list of segment numbers that are valid
goodlist=find(Dv(:,3));
%Find number of valid segments in Dv
nsegs=length(goodlist);
%Will hold center locations of LSE windows in d

dlocs=[];
```

```
%Will hold analogous center locations of LSE windows in s, according to
%the variable delay estimate
slocs_var=[];
%Loop over all segments
for i=1:nsegs
    %If it is the first segment in Dv
    if goodlist(i)==1
        %Starting sample number must be 1
        start=1;
    else
        %Otherwise it is 1 more than last sample of previous segment
        start=Dv(goodlist(i)-1,1)+1;
    end
    %Find last sample of segment
    stop=Dv(goodlist(i),1);
    %Find delay of the segment
    segdel=Dv(goodlist(i),2);
    %Find center of segment
    center=round((stop+start)/2);
    %Total number of LSE windows that will fit on this segment is 2*hn+1
    hn=floor((stop-center-320-lsewin/2)/maxsp);
    %Calculate the window location(s)
    if hn>=1
        locs=center+[-hn:1:hn]'*maxsp;
    else
        locs=center;
    end
    %Append locations to the list of LSE window center locations in d
    dlocs=[dlocs;locs];
    %Append locations to the list of LSE window center locations in s,
    %compensate for segment delay
    slocs_var=[slocs_var;locs-segdel];
end

%Create list of corresponding centers of LSE windows in s,
%according to the fixed delay estimate
```

```
slocs_fxd=dlocs-Df;

%Find 4 constants
L=round(lsewin/2);
R=lsewin-L-1;
len_d=length(d);
len_s=length(s);

%Find locations of window centers that will result in windows that do
%not extend beyond the ends of s or d
goodlocs=find( 1<=(dlocs-L) & (dlocs+R)<=len_d & 1<=(slocs_var-L) & ...
    (slocs_var+R)<=len_s & 1<=(slocs_fxd-L) & (slocs_fxd+R)<=len_s);
%Retain only such locations
dlocs=dlocs(goodlocs);
slocs_var=slocs_var(goodlocs);
slocs_fxd=slocs_fxd(goodlocs);
nlocs=length(dlocs);

%If there are locations for LSE calculations
if 0<nlocs
    %Build matrices of speech samples from the desired locations
    %Each column contains speech samples for a given LSE window
    D=zeros(lsewin,nlocs);
    Sv=zeros(lsewin,nlocs);
    Sf=zeros(lsewin,nlocs);
    for i=1:nlocs
        D(:,i)=d(dlocs(i)-L:dlocs(i)+R);
        Sf(:,i)=s(slocs_fxd(i)-L:slocs_fxd(i)+R);
        Sv(:,i)=s(slocs_var(i)-L:slocs_var(i)+R);
    end

    %Generate column vector with periodic Hanning window, length is lsewin
    win=.5*(1-cos(2*pi*[0:lsewin-1]'/lsewin));
    %Repeat this window in each column of the matrix Win.
    %(Win is lsewin by nlocs.)
    Win=repmat(win,1,nlocs);
    %Multiply by window and peform FFT on each column of each speech matrix
    D=fft(D.*Win);
    Sf=fft(Sf.*Win);
    Sv=fft(Sv.*Win);

    %Extract magnitude of unique half of FFT result
    D=abs(D(1:(lsewin/2)+1,:));
    Sf=abs(Sf(1:(lsewin/2)+1,:));
    Sv=abs(Sv(1:(lsewin/2)+1,:));

    %Limit results below at 1 to prevent log(0).  This is below the 10 dB
    %clamping threshold used below, so these clamped samples will not be
    %used
    D=max(D,1);
    Sf=max(Sf,1);
    Sv=max(Sv,1);

    %Take log and clamp results below at 10 dB.  Speech peaks will
    %typically be around 100 dB, so this limits dynamic range to about 90
    %dB and prevents low level segments from inappropriately dominating the
    %LSE results
    D=max(10,20*log10(D) );
    Sf=max(10,20*log10(Sf) );
    Sv=max(10,20*log10(Sv) );

    %Calculated LSE: inner mean is across frequency, outer mean is across
    %LSE windows (i.e. across time)
    lse_f=mean(mean(abs(D-Sf)));
    lse_v=mean(mean(abs(D-Sv)));

%LSE calculations are not possible
else
    lse_f=0;
    lse_v=0;
end
```

```
%==========================================================================
function SDV=median_filter(DCAVS,twinlen,winstep,activity_th,cor_th);
%Usage: SDV=median_filter(DCAVS,twinlen,winstep,activity_th,cor_th);
%This function does the median filtering on the results in the DCAVS
%matrix.  The DCAVS matrix is defined in the delay_tracking function.
%
%twinlen is the length of the median filtering window in ms
%winstep tell how many ms each step in DCAVS matrix corresponds to worth
%activity_th is the activity threshold required for a sample to be included
%   in the median filter
%cor_th is the correlation threshold required for a sample to be for
%   included in the median filtering
%
%Results are returned in the matrix SDV.  Each row of SDV describes a
%segment of constant delay.  SDV is in the fs=8000 domain.
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimate (0 for invalid, 1 for valid)
%
%SDV(end,1) is given by the center of the final window (DCAVS(end,5)),
%converted from the fs=500 domain to the fs=8000 domain.  Note that in
%general, this will not be exactly the same as the length of the
%speech signal y (output from system under test).
%
%If no valid information can be extracted from DCAVS, the result is a delay
%of zero everywhere.

%Extract number of time samples available
nwins=size(DCAVS,1);

%Create a temporary matrix.  It has 1 row for each row of DCAVS.
%Column 1 will hold delay estimates
%Column 2 will hold is 1 if that estimate is valid, 0 otherwise
%Column 3 will hold the distorted speech envelope sample number
%(in the fs=500 domain) associated with the delay estimate
T=[zeros(nwins,2),DCAVS(:,5)];
```

```
%Find half-width of median filtering window in samples
htwinlen=round(twinlen/(2*winstep));

%Good is a column vector with length nwins.  It has a 1 where a
%delay estimate has correlation that meets or exceeds threshold,
%activity that meets or exceeds threshold, and is mathematically valid.
%It has a zero elsewhere.
good=(cor_th<=DCAVS(:,2) & activity_th<=DCAVS(:,3) & DCAVS(:,4));

%Loop over all samples
for i=1:nwins
    %Check number of samples between sample and last sample
    nsmp=min(i-1,nwins-i);
    %Find final half-width of median filtering window (cannot exceed the
    %number of remaining samples)
    fhtwinlen=min(htwinlen,nsmp);
    start=i-fhtwinlen;              %First sample in current window
    stop=i+fhtwinlen;               %Last sample in current window
    %If there is a least one good sample in the window
    if sum(good(start:stop))>=1;
        %Form list of absolute indices of good samples
        goodlist=start+find(good(start:stop))-1;
        %Perform median filtering on the good samples
        %Note on median function:  When presented with an even number of
        %samples, this median function returns the average of the two
        %central samples.  e.g. median([1 2 3 4])=2.5
        T(i,1)=median(DCAVS(goodlist,1));
        %Mark that a valid result has been calculated
        T(i,2)=1;
    end
end

%Find how many valid results have been calculated
nvalres=sum(T(:,2));
```

```
%If no valid results have been calculated, report zero delay everywhere
if nvalres==0
    SDV=[DCAVS(end,5),0,0];
else
   %List only results that describe a change in delay or a change in
   %validity, plus the final result
   keepers=[find( (diff(T(:,1))~=0) | (diff(T(:,2))~=0)  );nwins];
   %Keep only those results
   SDV=T(keepers,[3 1 2]);
end

%Convert final results from fs=500 samples/sec domain to the
%fs=8000 samples/sec domain
SDV(:,2)=SDV(:,2)*16;                       %Convert delay estimates
SDV(:,1)=(SDV(:,1)-1)*16+1+8;               %Convert sample values
```

```
%===========================================================================
function [xcs,denom,ystart,ystop]=non_fft_xc(x,y,min_d,max_d)
%Usage: [xcs,denom,ystart,ystop]=non_fft_xc(x,y,min_d,max_d)
%This function enables delay estimation by calculating the cross
%correlation between two vectors of speech samples x and y at the
%specified shifts. x and y need not have the same length.  Zero delay
%is associated with the case where x(1) aligns with y(1). x and y may
%be row or column vectors.  Cross correlations are performed at all
%delays between min_d and max_d (given in samples) inclusive.  If the
%length of x or y prevents this, an error is generated.
%xcs is a column vector and holds unnormalized correlation values for all
%   requested delays
%denom is the normalization factor so that xcs/denom will be true
%correlation values.
%Note that length(xcs)=max_d-min_d+1. xcs(1) is associated with min_d,
%xcs(end) is associated with max_d.
%Note also that this function always uses a fixed segement of samples of y.
%The number of samples in this fixed segment is maximized given the
%constraints imposed by the lengths of x and y as well as the values of
%min_d and max_d.
%ystart and y stop are the first and last samples of y that are used in
%the fixed segment.

%Find number of shifts
nshifts=max_d-min_d+1;

%Initialze correlation results variable
xcs=zeros(nshifts,1);

%Find two lengths
nx=length(x);
ny=length(y);

%Find segment of y that can be used
ystart=max(1,max_d+1);   %ystart is first sample of y to use
if ny - min_d <= nx
```

```
    ystop=ny;           %ystop is last sample of y to use
else
    ystop=nx+min_d;
end

%Generate error if there is no useable segment of y
if ystop<ystart
    error('Not enough input samples to calculate all delay values.')
end

%Extract segment of y
temp_y=y(ystart:ystop);

%Find number of samples in segment
m=ystop-ystart+1;

%Loop over all shifts
for i=1:nshifts
    %Update current delay value
    cd=min_d+i-1;
    xstart=ystart-cd;

    %Extract proper segment of x
    temp_x=x(xstart:xstart+m-1);

    %Form partial denominator of correlation so it can be tested
    %to prevent divide by zero
    denom=temp_x'*temp_x;
    if denom>0
        %Correlation
        xcs(i)=(temp_x'*temp_y)/sqrt(denom);
    else
        %When segment of x has no signal, correlation is zero
        xcs(i)=0;
    end
```

```
end

%Find the fixed portion of the denominator
denom=sqrt(temp_y'*temp_y);
```

```
%============================================================================
function [un_corr,denom]=non_fft_xc_all(x,y)
%This function applies a direct-form cross correlation to the column
%vectors of speech samples x and y.  This correlation is done for all
%possible shifts that use all of y.
%The unnormalized correlation values are returned in un_cor, and the
%denominator is returned in denom.
%It is required that length(x)>=length(y).

%Find lengths
nx=length(x);
ny=length(y);

%Find number of possible shifts
nshifts=nx-ny+1;

%Initialize results variable
un_corr=zeros(nshifts,1);

%Loop over all possible shifts
for i=1:nshifts
    temp_x=x(i:i+ny-1);
    %Form partial denominator of correlation so it can be tested
    %to prevent divide by zero
    denom=temp_x'*temp_x;
    if denom>0
        %Correlation
        un_corr(i)=(temp_x'*y)/sqrt(denom);
    else
        %When segment of x has no signal, correlation is zero
        un_corr(i)=0;
    end
end
%Find fixed portion of denominator
denom=sqrt(y'*y);
```

```
%==========================================================================
function SDVout=short_seg_cor(SDVin,x_speech,y_speech,len_t,len_b,len_s)
%Usage:  SDVout=short_seg_cor(SDVin,x_speech,y_speech,len_t,len_b,len_s)
%This function tests all pulses (also called blips), steps and tails in
%an estimated delay history and removes them when appropriate
%
%x_speech and y_speech are are column vectors that hold system under test
%   input and output speech samples (without any delay compensation)
%len_t is the length in ms of the longest tail that should be removed
%len_b is the length in ms of the longest blip that should be removed
%len_s is the length in ms of the longest step that should be removed
%SDVin and SDVout are Delay history matricies in the fs=8000 domain.  There
%is one row per segment of constant delay
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimation for segment
%           (0 for invalid, 1 for valid)

%Convert from ms to samples
len_t=8*len_t;
len_b=8*len_b;
len_s=8*len_s;

%Find number of segments of constant delay
nsegs=size(SDVin,1);

%Find length of each segment
seglens=diff([0;SDVin(:,1)]);

%Append two columns to SDVin.  SDVin now has
%Column 1, Sample number of last sample of constant delay segment
%Column 2, estimated Delay of segment
%Column 3, Validity of delay estimation segment (0=invalid, 1=valid)
%Column 4, Number of samples in segment
%Column 5, Status of segment (0=needs to be considered, 1=should be
%ingnored)  Start with all zeros.
```

```
SDVin=[SDVin,seglens,zeros(nsegs,1)];

%Find location and type of the shortest segment with status 0 in SDVin
[ptr,seg_type]=find_smallest_seg(SDVin);

%If there is such a segment
if ptr~=0;
    %Extract its length
    current_seg_len=SDVin(ptr,4);
else
    %Otherwise create a fictitious segment length that is long enough
    %to prevent entering the "while loop" that follows
    current_seg_len=max([len_t len_b len_s])+1;
end

%While the shortest segment qualifies for consideration under at least
%one of the three length thresholds
while current_seg_len<=max([len_t len_b len_s])
    %Find number of segments in current version of SDVin
    n=size(SDVin,1);
    %If current segment is a left tail and conforms with tail threshold
    if seg_type=='LT' & current_seg_len <= len_t
        %Join current segment to right neighbor segment
        %to create new combined segment
        SDVin=SDVin(setdiff(1:n,ptr),:);
        %(Setdiff is the set difference function.  As called above, it
        %returns a length n-1 vector containing 1,2,...ptr-1, ptr+1, ...n)
        %Set status on the new combined segment to 0 so it receives
        %further consideration
        SDVin(ptr,5)=0;
        %Store the length of the new combined segment
        SDVin(ptr,4)=SDVin(ptr,4)+current_seg_len;
    %If current segment is a right tail and conforms with tail threshold
    elseif seg_type=='RT'  & current_seg_len <= len_t
        %Join current segment to left neighbor segment
        %to create new combined segment
        current_seg_end=SDVin(ptr,1);
        SDVin=SDVin(setdiff(1:n,ptr),:);
        SDVin(ptr-1,1)=current_seg_end;
        %Set status on the new combined segment to 0 so it receives
        %further consideration
        SDVin(ptr-1,5)=0;
        %Store the length of the new combined segment
        SDVin(ptr-1,4)=SDVin(ptr-1,4)+current_seg_len;
    %If current segment is a blip and conforms with blip threshold
    elseif seg_type=='BI'  & current_seg_len <= len_b
        %Join current segment and left neighbor to right neighbor segment
        %to create new combined segment
        left_neb_len=SDVin(ptr-1,4);
        SDVin=SDVin(setdiff(1:n,[ptr-1 ptr]),:);
        %Set status on the new combined segment to 0 so it receives
        %further consideration
        SDVin(ptr-1,5)=0;
        %Store the length of the new combined segment
        SDVin(ptr-1,4)=SDVin(ptr-1,4)+current_seg_len+left_neb_len;
    %If current segment is a step and conforms with step threshold
    elseif seg_type=='SP'  & current_seg_len <= len_s

        %Join current segment to left or right neighbor or leave it
        %as is.  Choice of these 3 actions depends on correlation results

        %----------------------Preparations------------------------------
        start=SDVin(ptr-1,1)+1;%First sample of current segment
        stop=SDVin(ptr,1);     %Last sample of current segment
        L_dly=SDVin(ptr-1,2);  %Delay of segment left of current segment
        C_dly=SDVin(ptr,2);    %Delay of current segment
        R_dly=SDVin(ptr+1,2);  %Delay of segment right of current segment

        %Correlation at delay of left neighbor
        lcorr=single_corr(x_speech,y_speech,start,stop,L_dly);
        %Correlation at delay of right neighbor
        rcorr=single_corr(x_speech,y_speech,start,stop,R_dly);
        %Correlation at delay of current segment
        ccorr=single_corr(x_speech,y_speech,start,stop,C_dly);

        %Which of these 3 correlations is largest?
        [dud,loc]=max([lcorr rcorr ccorr]);
        %If correlation at delay of left neighbor is largest
        if loc==1
            %Join current segment to left neighbor segment
            %to create new combined segment
            current_seg_end=SDVin(ptr,1);
            SDVin=SDVin(setdiff(1:n,ptr),:);
            SDVin(ptr-1,1)=current_seg_end;
            %Set status on the new combined segment to 0 so it receives
            %further consideration
            SDVin(ptr-1,5)=0;
            %Store the length of the new combined segment
            SDVin(ptr-1,4)=SDVin(ptr-1,4)+current_seg_len;
        %If correlation at delay of right neighbor is largest
        elseif loc==2
            %Join current segment to right neighbor segment
            %to create new combined segment
            SDVin=SDVin(setdiff(1:n,ptr),:);
            %Set status on the new combined segment to 0 so it receives
            %further consideration
            SDVin(ptr,5)=0;
            %Store the length of the new combined segment
            SDVin(ptr,4)=SDVin(ptr,4)+current_seg_len;
        else
            %Don't change the step, but change its status to 1 for no
            %further consideration
            SDVin(ptr,5)=1;
        end
    %For all other segment types, this function makes no changes.
    else
        %Change status to 1 for no further consideration
        SDVin(ptr,5)=1;
    end

    %Find location and type of the shortest segment with status 0 in SDVin
    [ptr seg_type]=find_smallest_seg(SDVin);

    %If there is such a segment
    if ptr~=0;
        %Extract its length
        current_seg_len=SDVin(ptr,4);
    else
        %Otherwise create a fictitious segment length that is long enough
        %to terminate the "while loop"
        current_seg_len=max([len_t len_b len_s])+1;
    end

%End of while loop
end

%SDVout contains just the first 3 columns of SDVin
SDVout=SDVin(:,1:3);
%Identify the segments that reflect a change in delay or validity
keepers=[find(  (diff(SDVout(:,2))~=0) |  ...
                                   (diff(SDVout(:,3))~=0));size(SDVout,1)];
%Retain only those segments
SDVout=SDVout(keepers,:);
```

```
%===========================================================================
function rho=single_corr(x_speech,y_speech,start,stop,delay)
%Usage:   rho=single_corr(x_speech,y_speech,start,stop,delay)
%This function calculates a single correlation value between
%a segment of x_speech and y_speech.  The goal is to use the
%samples of y_speech from "start" to "stop" inclusive and the
%corresponding portion of x_speech, but shifted forward in time
%by the number of samples specified in "delay."
%These segments may be shortened as necessary if not enough
%samples are available.
%The correlation is direct-form and is performed by the
%function non_fft_xc_all

%First sample of x_speech that will be used
sstart=start-delay;
lefttrim=0;
%If that sample does not exist
if sstart<1
    %It is necessary to shift the start of the correlation window
    lefttrim=1-sstart;
    sstart=1;
end
%Last sample of x_speech that will be used
sstop=stop-delay;
ns=length(x_speech);
righttrim=0;
%If that sample does not exist
if sstop>ns
    %It is necessary to shift the end of the correlation window
    righttrim=sstop-ns;
    sstop=ns;
end
%Extract speech from correlation window and rectify
x=abs(x_speech(sstart:sstop));
y=abs(y_speech(start+lefttrim:stop-righttrim));
%Perform unnormalized correlation
```

```
[rho,denom]=non_fft_xc_all(x,y);
%Normalize to find correlation value
rho=rho/denom;
```