SRI International

23 December 2002

IMPROVED OPEN-MICROPHONE SPEECH RECOGNITION

FINAL REPORT

Grant No: NAG 2-1568

Period of Performance: 6/28/02 – 12/15/02

Prepared for: NASA Ames Research Center
Sharon L. Connolly
Grant Office, Mail Stop 241-1
Moffett Field, CA 94035-1000

SUBMITTED BY

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025-3493

	Principal Investigator	Administrative Contact	Financial Contact
Name:	Victor Abrash	Margaret Baxter-Pearson	Barbara Guthary
Phone:	(650) 859-4490	(650) 859-4424	(650) 859-2903
Fax:	(650) 859-5984	(650) 859-6171	(650) 859-5984
Email:	victor@speech.sri.com	margaret.baxter-pearson@sri.com	barbara@speech.sri.com

The material is based upon work supported by NASA under Award No. NAG 2-1568. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

INTRODUCTION

Many current and future NASA missions make extreme demands on mission personnel both in terms of work load and in performing under difficult environmental conditions. In situations where hands are impeded or needed for other tasks, eyes are busy attending to the environment, or tasks are sufficiently complex that ease of use of the interface becomes critical, spoken natural language dialog systems offer unique input and output modalities that can improve efficiency and safety. They also offer new capabilities that would not otherwise be available [1]. For example, many NASA applications require astronauts to use computers in micro-gravity or while wearing space suits. Under these circumstances, command and control systems that allow users to issue commands or enter data in hands- and eyes-busy situations become critical [2].

Speech recognition technology designed for current commercial applications limits the performance of the open-ended state-of-the-art dialog systems being developed at NASA. For example, today's recognition systems typically listen to user input only during short segments of the dialog, and user input outside of these short time windows is lost. Mistakes detecting the start and end times of user utterances can lead to mistakes in the recognition output, and the dialog system as a whole has no way to recover from this, or any other, recognition error. Systems also often require the user to signal when that user is going to speak, which is impractical in a hands-free environment, or only allow a system-initiated dialog requiring the user to speak immediately following a system prompt.

In this project, SRI has developed software to enable speech recognition in a hands-free, open-microphone environment, eliminating the need for a push-to-talk button or other signaling mechanism. The software continuously captures a user's speech and makes it available to one or more recognizers. By constantly monitoring and storing the audio stream, it provides the spoken dialog manager extra flexibility to recognize the signal with no audio gaps between recognition requests, as well as to rerecognize portions of the signal, or to rerecognize speech with different grammars, acoustic models, recognizers, start times, and so on. SRI expects that this new open-mic functionality will enable NASA to develop better error-correction mechanisms for spoken dialog systems, and may also enable new interaction strategies.

DELIVERABLE

SRI has developed a custom "continuous" audio provider compatible with the Nuance 7 and Nuance 8 speech recognition systems. "The audio provider is the part of the recognition client that manages the connections to the audio input and output devices and delivers audio samples to the Nuance recognition server for speech recognition." [3]

The continuous audio provider implements standard playback and recording functionality on Windows 2000 PC platforms with a full-duplex audio card and Microsoft's DirectX Application Programming Interface (API) installed. In addition to standard recording, the continuous audio provider offers the option to record audio even when recognition is not active. The continuous audio provider also offers the option to send audio simultaneously to two recognizers, and can return measurements of audio signal quality to the speech application.

The software utilizes the DirectSound API available from Microsoft. This is the most up-to-date sound interface available for Windows-based personal computers, and minimizes both recording and playback latency and potential problems with other audio components of the speech dialog system,

such as text-to-speech (TTS) synthesis. Because it uses DirectSound, the software may not be compatible with Windows NT or older versions of Windows 95, and may require installing or updating Microsoft DirectX on computers on which systems are deployed.

The software is implemented as a Nuance custom Audio Provider (i.e., it implements the Nuance AudioProviderInfo interface); therefore, it can be used as a plug-in component with any Nuance RCEngine or RCAPI-based dialog application, or used with any other Nuance API that supports audio provider plug-ins. The Nuance endpointer can continue to be used in the recognition server. The continuous audio provider has been tested with and is compatible with the Nuance RCAPI and RCEngine APIs and the existing NASA OAA speech recognition agent.

FEATURES

Continuous Recording

In addition to standard recording, the continuous audio provider can record audio even when recognition is not active. This allows spoken dialog systems to recognize the speech signal with no audio gaps between recognition requests.

The continuous audio provider software essentially consists of a continuously recording audio "tape". The amount of speech stored within the audio provider (the length of the "tape") is configurable, and is limited only by the amount of computer memory available.

Because audio can be recorded even when the recognizer is not active, new interaction strategies become available. For example, speech recognition can be performed on the audio stream starting:

1. when recognition was started
2. from any specific time
3. from a specific time before now
4. from the last end-of-speech detection time
5. from the last start-of-speech detection time

The recognition mode is configurable at runtime, and can be changed between one utterance and the next. Currently, only modes (1) and (4) are implemented.

The recording mode is controlled via the "audio.continuous.RecordMode" or "audio.continuous.RecognitionMode" parameters. These two parameters perform exactly the same function and can be used interchangeably. They determine which samples are returned to the Nuance RecClient during recognition or recording.

If the value is set to "on-next-start-recognition", only audio samples occurring after recognition or recording was started are sent to the recognizer. Setting the mode to this value has the same effect as disabling continuous recording since samples recorded before recording/recognition was started are discarded. The recognizer will perform exactly as if the "native" audio provider were being used.

The continuous recording mode is specified by setting the value of "audio.continuous.RecordMode" to "on-previous-end-recognition". In this mode, all audio samples recorded since recognition or recording was last stopped are sent to the recognizer.

When using the continuous recording mode, the additional parameter "audio.continuous.ToggleRecordMode" is used to turn continuous recording on or off. Setting its value to "onrecognize" or "start" enables continuous recording. "stop" ends and disables continuous recording but saves any samples already recorded. "abort" ends and disables continuous recording and discards any previously recorded samples.

If "audio.continuous.ToggleRecordMode" is set to "start" during initialization or via a "SetParameter()" call, then continuous recording will start immediately.

If "audio.continuous.ToggleRecordMode" is set to "onrecognize" (the default) during initialization or via a "SetParameter()" call, then continuous recording will start the next (or first) time that recognition or recording is started.

If "audio.continuous.ToggleRecordMode" is set to "stop" or "abort" during initialization, then continuous recognition will be disabled. Setting this parameter to "stop" or "abort" via a "SetParameter()" call will immediately halt continuous recording, and save or discard previously recorded samples as described above.

Simultaneous Recognition with Two Recognizers

The continuous audio provider can send audio simultaneously to two recognizers.

This capability is controlled by the "audio.continuous.MultiChannelMode" parameter.

If the value is set to "disable" (the default), then audio is sent only to a single recognizer. Any number of recognizers can be created in this mode.

Setting "audio.continuous.MultiChannelMode" to "master" enables the audio provider to record audio and send it to its own recognizer plus another recognizer whose audio provider has "audio.continuous.MultiChannelMode" set to "slave".

Setting the value to "slave" disables audio recording, and configures the audio provider to receive audio from the "master" audio provider rather than the microphone.

Only one audio provider should be initialized as a "master", and only one other audio provider should be initialized as a "slave". The "master" recognizer should be created first. The "slave" recognizer cannot play audio.

Signal Analysis

The custom audio provider can provide several measurements of audio signal quality to the speech application. At the end of each utterance, the following measurements can be retrieved as a string using the "audio.continuous.AudioStatus" parameter:

- signal-to-noise ratio (SNR)
- root-mean-square energy (RMS) of the combined speech and noise signal
- RMS of the noise signal
- minimum and maximum sample values
- number of samples clipped (absolute value > 31000)

- duration (in seconds)
- DC offset

USING THE CONTINUOUS AUDIO PROVIDER

To use the continuous audio provider, start the recognition client with the Nuance "audio.Provider" parameter set to "continuous". To enable continuous recording, set the parameter "audio.continuous.RecordMode" to "on-previous-end-recognition".

The speech recognition application will also need access to the compiled continuous audio provider "apcontinuous.dll", which should be located local to the recognition client or within the current path.

MANUAL PAGES

Before using the continuous audio provider, please read the "Continuous Audio Provider Documentation" provided with this report.

REQUIREMENTS

The following hardware and software is required to run the continuous audio provider:

- Microsoft Windows 2000 PC with DirectX 7 or greater installed
- Nuance 7 or 8 speech recognizer
- Sound card, speakers, and microphone for the PC
- An application based on Nuance's RCEngine, RCAPI, or other APIs which can load the audio provider

These additional components are required to build the continuous audio provider:

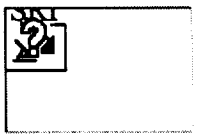
- Microsoft Visual Studio 6.0
- Nuance 7 or 8 libraries
- The DirectX 7 or greater DirectSound library (dsound.lib) 8.1

REFERENCES

[1] RIALIST FY01 Strategic Plan, http://www.riacs.edu/research/detail/rialist.

[2] "Do That Again": Evaluating Spoken Dialogue Interfaces, F. James, M. Rayner, and B.A. Hockey, 2000, RIACS Technical Report #00.06.

[3] "Nuance Platform Integrator's Guide", Nuance Communications.



Continuous Audio Provider Documentation

Introduction

The continuous audio provider implements standard playback and recording functionality on Windows PC platforms using the Microsoft's DirectX API. In addition to standard recording, the continuous audio provider offers the option of continuously recording audio even when recognition is not active. The continuous audio provider also offers the option to send audio simultaneuously to two recognizers.

The continuous audio provider implements the Nuance AudioProviderInfo interface so it can be used as a plug-in component with any RCEngine, or RCAPI-based application, or with any other API that supports audio provider plug-ins.

Continuous Recording

Since recording can continue after recognition has stopped, there are a variety of continuous recording options. Configure these options by enabling or disabling continuous recording using the `audio.continuous.ToggleRecordMode` parameter. Then use the `audio.continuous.RecognitionMode` or `audio.continuous.RecordMode` parameter to determine which samples captured during continuous recording are returned during recognition or recording.

The values for the continuous recording parameters are summarized below in the Audio Configuration Parameters table, but in general these continuous recording parameters have two purposes.

First, `audio.continuous.ToggleRecordMode` is used to indicate whether or not continuous recording is enabled. If continuous recording is disabled, then recording occurs only when the Nuance client is recognizing or recording; otherwise, the audio provider records continuously.

Second, `audio.continuous.RecognitionMode` (or `audio.continuous.RecordMode`) indicates which samples are returned to the client during recognition/recording. If the continuous recording is disabled, then there are no recognition/recording options; only audio recorded during the period the Nuance client is actively recognizing or recording is returned to the client. On the other hand, if continuous recording is enabled then there are many options for selecting audio that is returned for recognition/recording.

When the continuous audio provider is set for continuous recording then it is possible to return audio recorded at a specific time before, after, or since

- Recognition was started by the client
- Recognition was previously ended by the client
- Recording was started by the continuous audio provider

- End-of-speech was detected
- Start-of-speech was detected

Not all of the above options are implemented. See the Audio Configuration Parameters table below for a complete listing of the modes currently supported by the continuous audio provider.

Sending Audio to Multiple Recognizers

The continuous audio provider also offers the option to send audio simultaneuously to two recognizers. This capability is controlled by the `audio.continuous.MultiChannelMode` parameter. If the value is set to "disable" (the default), then audio can be sent to only a single recognizer. Setting `audio.continuous.MultiChannelMode` to "master" enables the audio provider to record audio and send it to its own recognizer plus another recognizer with a `audio.continuous.MultiChannelMode`=slave audio provider. Setting the value to "slave" disables audio recording, and configures the audio provider to receive audio from the "master" audio provider. Only one audio provider should be initialized as a "master", and only one other audio provider should be initialized as a "slave". The "master" recognizer should be created first. The "slave" recognizer cannot play audio.

Signal Analysis

The custom audio provider can provide several measurements of audio signal quality to the speech application. At the end of each utterance, the following measurements can be retrieved as a string using the `audio.continuous.AudioStatus` parameter:

- signal-to-noise ratio (SNR)
- root-mean-square energy (RMS) of the combined speech and noise signal
- RMS of the noise signal
- minimum and maximum sample values
- number of samples clipped (absolute value > 31000)
- duration (in seconds)
- DC offset

Requirements

The following hardware and software is required to run the continuous audio provider:

- Microsoft Windows PC with DirectX 7 or greater installed
- Nuance 7 or 8 speech recognizer
- Sound card, speakers, and microphone for the PC
- An application based on Nuance's RCEngine, RCAPI, or other APIs that can load the audio provider

These additional components are required to build the continuous audio provider:

- Microsoft Visual Studio 6.0

- Nuance Verifier 2.0 or later
- The DirectX 7 or greater DirectSound library (dsound.lib) 8.1

Configuration

These parameters may be used to configure the audio provider:

Parameter Name	Runtime Set	Runtime Get	Type	Default
audio.Provider	No	Yes	String	native
audio.format	Yes	Yes	String	8k-8bit-mulaw
audio.InputVolume	Yes	Yes	Int	200
audio.continuous.InputVolume	Yes	Yes	Int	
audio.continuous.MasterInputVolume	Yes	Yes	Int	
audio.continuous.InputMute	Yes	Yes	bool	
audio.continuous.MasterInputMute	Yes	Yes	bool	
audio.OutputVolume	Yes	Yes	Int	
audio.continuous.OutputVolume	Yes	Yes	Int	200
audio.continuous.MasterOutputVolume	Yes	Yes	Int	
audio.continuous.OutputMute	Yes	Yes	bool	
audio.continuous.MasterOutputMute	Yes	Yes	bool	
audio.continuous.NumberOfPlaybackSegments	No	Yes	String	4
audio.continuous.PlaybackBufferSecs	No	Yes	Int	2
audio.continuous.CaptureBufferSecs	No	Yes	Int	2
audio.continuous.AudioContainerSecs	No	Yes	Int	5
audio.continuous.GlobalPlayFocus	Yes	Yes	Int	1
audio.continuous.AudioStatus	No	Yes	String	--
audio.continuous.RecordMode	No	Yes	String	on-next-start-recoc
audio.continuous.RecognitionMode	No	Yes	String	on-next-start-recoc
audio.continuous.ToggleRecordMode	Yes	Yes	string	onrecognize
audio.continuous.MultiChannelMode	No	Yes	String	disable

audio.continuous.DebugLevel	Yes	Yes	Int	4
audio.continuous.RecordMsecsDropped	No	Yes	Int	0
audio.continuous.MaxSpeechSec	N/A	N/A	N/A	N/A

`audio.Provider`

Specifies the audio device to use for live audio recording and playback. Set the value to "continuous" to use the continuous audio provider.

`audio.format`

Specifies the audio sampling rate, precision, and encoding, as described in the Nuance manual. SRI recommends "8K-16bit-linear" rather than "8K-8bit-mulaw".

`audio.InputVolume`

Sets the audio input volume on a machine-independent scale of 0 to 255.

`audio.continuous.InputVolume`

Same as `audio.InputVolume`.

`audio.continuous.MasterInputVolume`

Same as `audio.InputVolume`.

`audio.continuous.InputMute`

Mute or un-mute the input. When mute is set to 1, the volume level is set to 0. When mute is set to 0, the previous volume level is restored.

`audio.continuous.MasterInputMute`

Same as `audio.continuous.InputMute`.

`audio.OutputVolume`

Sets the audio output volume on a machine-independent scale of 0 to 255.

`audio.continuous.OutputVolume`

Same as `audio.OutputVolume`.

`audio.continuous.MasterOutputVolume`

Same as `audio.OutputVolume`.

`audio.continuous.OutputMute`

Mute or un-mute the output. When mute is set to 1, the volume level is set to 0. When

mute is set to 0, the previous volume level is restored.

`audio.continuous.MasterOutputMute`

Same as `audio.continuous.OutputMute`

`audio.continuous.NumberOfPlaybackSegments`

Number of segments in the DirectX SoundBuffer used for playback. Similar to `audio.native.NumberOfAudioBuffers`, but applies only to the playback buffer and not the capture buffer. This value cannot be set at runtime.

`audio.continuous.PlaybackBufferSecs`

Length in seconds of the DirectX SoundBuffer used for playback. This is similar to the `audio.native.InputChunkSeconds` parameter, but this parameter describes the size of the entire playback buffer and not individual segments. This value cannot be set at runtime.

`audio.continuous.CaptureBufferSecs`

Length in seconds of the DirectX CaptureBuffer used for recording. The capture buffer is not divided into segments, so this parameter describes the size of the entire capture buffer. This value cannot be set at runtime.

`audio.continuous.AudioContainerSecs`

Length in seconds of the internal audio structure used to store samples for playback and recording. This value cannot be set at runtime.

`audio.continuous.GlobalPlayFocus`

If set to zero, playback will stop when the application window loses focus; otherwise, playback will continue. By default, playback will continue. This value can be set at runtime.

`audio.continuous.AudioStatus`

Retrieves the audio status for the last recognized or recorded utterance as a string. The values include SNR, RMS, RMS(noise window), minimum and maximum sample values, the number of samples clipped (value > 31000), the duration (in seconds), and the DC offset.

`audio.continuous.RecordMode`

Determines which samples are returned to the client when recording or recognition is started.

- `on-next-start-recognition`: only audio samples recorded since recognition or recording was started are returned. Setting the mode to this value has the same effect as disabling continuous recording since samples recorded before recording/recognition was started are discarded.

- `on-previous-end-recognition`: returns all audio samples recorded since recognition or recording was last stopped. This attribute is used in conjunction with the `audio.continuous.ToggleRecordMode` attribute.

`audio.continuous.RecognitionMode`

An alias for the `audio.continuous.RecordMode` attribute

`audio.continuous.ToggleRecordMode`

Controls continuous recording. "onrecognize" and "start" enable continuous recording. "stop" ends and disables continuous recording but saves any samples already recorded. "abort" ends and disables continuous recording and discards any previously recorded samples.

If you set this attribute to "start" during initialization or via a `SetParameter()` call, then continuous recording will start immediately.

If you set this attribute to "onrecognize" (the default) during initialization or via a `SetParameter()` call, then continuous recording will start the next time that recognition or recording is started.

If you set this attribute to "stop" or "abort" during initialization, then continuous recognition will be disabled. Setting this attribute to "stop" or "abort" via a `SetParameter()` call will imediately halt continuous recording, and save or discard previously recorded samples as described above.

Use this attribute in conjunction with the `audio.continuous.RecognitionMode` and `audio.continuous.RecordMode` attributes

`audio.continuous.MultiChannelMode`

Controls sending audio to multiple recognizers.

If the value is set to "disable" (the default), then audio can be sent only to a single recognizer. Any number of recognizers can be created with this value.

"master" enables the audio provider to record audio and send it to its own recognizer plus another recognizer with a "slave" audio provider. "slave" disables audio recording, and configures the audio provider to receive audio from the "master" audio provider.

Only one audio provider should be initialized as a "master", and only one other audio provider should be initialized as a "slave". The "master" recognizer should be created first. The "slave" recognizer cannot play audio.

`audio.continuous.RecordMsecsDropped`

The number of milliseconds of audio data dropped during the last recording session. This value is not currently set, and always returns 0

`audio.continuous.DebugLevel`

Not implemented.

`audio.continuous.MaxSpeechSec`

Not implemented. This parameter could determine the maximum amount of audio saved during continuous recording.

Using the Continuous Audio Provider

To use the continuous audio provider, execute the recognition client with the audio.Provider parameter set to continuous (audio.Provider=continuous). If you want continuous recording, then you should also set the parameter `audio.continuous.RecordMode` to `on-previous-end-recognition`.

Your Nuance application will also need access to apcontinuous.dll, which should be located local to the recognition client or within the current user's path.

Building the Continuous Audio Provider

Use the Microsoft Visual C++ workspace ContinuousAP7.dsw or ContinuousAP8.dsw to build the continuous audio provider for Nuance 7 and Nuance 8, respectively. The libraries distributed with Nuance and the SRI utilities library distributed with the continuous audio provider are required to complete the build.

The source code for the utilities library should not be shared outside NASA RIACS; only the compiled library should be sent to other sites.

Known Issues

1. In continuous recording mode, there may be long pauses in the recorded audio. The Nuance endpointer will detect these long pauses and endpoint on them.

For More Information

- Nuance - offers more information on Audio Providers, the RCEngine API, or other aspects of Nuance Speech Development. The Nuance Verifier may be downloaded from Nuance by joining the Nuance Developers Network.
- Microsoft DirectX - information on the DirectX API, including DirectSound, which is used to implement playback and recording for the continuous audio provider.

Last modified: Mon Dec 16 17:05:38 Pacific Standard Time 2002
File Version: $Id: Readme.html,v 1.11 2002/12/17 01:03:06 victor Exp $