HANDS-FREE HUMAN-MACHINE DIALOGUE
CORPORA, TECHNOLOGY AND EVALUATION

Frank K. Soong and Eric A. Woudenberg

Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey, USA

ABSTRACT

In this paper we will review the progress of hands-free, Voice User Interface (VUI) research work at Bell Labs, including: a multichannel data base collection, technology development, and performance evaluation. Thirty-channel, simultaneous recordings have been conducted in a moving car, collecting speech from 57 subjects under various weather, road, and noise conditions. These are being used for both testing and adaptation purposes. Technology issues relevant to hands-free VUI are specifically addressed, including: (1) acoustic echo cancellation (AEC) and near-end (user's) speech detection; (2) background noise estimation and suppression; (3) reliable and timely barge-in; (4) signal pickup improvement using intelligent microphone arrangements; and (5) speaker and environment adaptation. An evaluation of the developed technologies using the car database is presented. An all software, hands-free, full duplex voice user interface demo has been implemented on a LINUX PC. The real-time demo provides services like: voice-dialing (dialing a person by name or a connected digit string), information service (accessing headline news, weather reports, sports and stock quotations), personal message service (retrieving email, voice mail and fax) and voice control of a DVD-player (selecting topics, controlling volume and video playback speeds).

1. Introduction

Currently, automatic speech recognition is being deployed in many application areas such as voice dictation, key-word based automatic telephone operator assistance, automatic voice dialing and call routing, etc. However the applications, despite their popularity, are still rather limited and in many cases head-mounted, close-talking microphones or telephone handsets are required. To free users from these tethering constraints and to facilitate truly hands-free human machine communication, technologies need to be advanced. Ongoing research has been done in various organizations [7] [8] [11]. In this paper we give a progress report on our infrastructure development, technology advances, and system integration and evaluation at Bell Labs, Lucent Technologies. First, we will give a review of our database collection effort. A series of simultaneous 30-channel recordings from 57 subjects has been conducted in a Ford Explorer SUV under real driving conditions. The database is used for developing and evaluating many of the critical technologies for hands-free VUI applications. We next present our research advancement along various technology front-ends and its corresponding results. Finally, we discuss the issues relevant to LINUX PC implementation.

2. A Multi-Channel Car Database

Before we can engage in in-depth research work on hands-free VUI technologies, collecting a speech database which encompasses the rich acoustic varieties in such an environment is a prerequisite. A speech database collected in a car under real driving conditions, with multi-channel microphones, is a good starting point. Useful applications, e.g., hands-free voice control of various devices such as cellular phones, audio/video equipments, windows, wipers, etc., and hands-free access to information such as email, voice-mail and fax messages through wireless links make such a database highly desirable. Many unsolved problems such as robust ASR in the adverse acoustic environment of a moving car and adaptive beam-forming using multichannel input justify the effort to collect such a speech database.

Also, a voice-controlled, human-machine interface is considerably safer for a driver to operate than a keyboard coupled with a visual display. A database has thus been collected for research work in acoustic signal processing operations such as beamforming, noise suppression, and robust front-end feature selection as well as for ASR related tasks such as on-line noise tracking and modeling, and speaker and environment adaptation. Technologies so developed will find immediate relevance to in-vehicle speech applications.

Recording Channel Assignment

Thirty channels have been simultaneously recorded to Hi-8 video tape via a TEAC digital recorder sampling at 24 kHz with 16 bit precision. A list of these channels is given in Table 1. The linear microphone array, consisting of 16 equi-spaced, hyper-cardioid microphone elements and mounted in lieu of the sunvisor on the passenger side. For comparison purposes, an omnidirectional microphone was installed next to the linear array. Four additional microphones (two omni and two hyper-cardioid) were mounted on top of the dashboard. Two close-talking boom microphones were worn by the subject, and those recordings were used as high signal-to-noise ratio (SNR) reference signals. Since all channels were recorded in a time-aligned fashion, the speech endpoints found in the close-talking microphone channels were conveniently used to precisely segment all the other data channels as well. The close-talking signals additionally serve as an upper limit baseline measure of ASR performance.

The left and right stereo channels of the car radio were recorded as well, for stereo-echo cancellation study. The velocity and position of the car were recorded, as reported through the ASCII output of a roof mounted GPS receiver. An x-y-z 3-axis acceleration sensor was cemented to the windshield to record mechanical movements (up to 12 kHz) of the car in order to register any chassis vibration from e.g., uneven road conditions. This information can be used to distinguish acoustic noise from mechanical movement of the microphone's membrane. Finally, a separate channel recorded laptop ASCII data about the road type, weather condition, radio state, etc. as well as the text prompts that were displayed on the laptop screen for the subjects. As can be seen in Figure 1, the SNR's of simultaneous recordings from the close-talking, single array element, and omnidirectional microphones are significantly different. Due to the directivity of the hyper-cardioid microphones, noise from the windshield is significantly reduced and the recorded signal is much quieter than that from the omni-directional, but still noisier than its close-talking counterpart.
Figure 1: Speech signals recorded simultaneously from the close-talking (head-mounted), hyper-cardioid (visor-mounted, single element), and omni-directional (dashboard-mounted) microphones.

### Recording Material, Subjects and Conditions

In order to cover a wide range of applications, the content of the database has been designed to include:

- Control commands for car applications
- Digit strings of different length ranging from single, isolated digits to 7-digit strings
- City names, company names and commands for car navigation systems
- TIMIT phonetically rich sentences

The database consists of 87 predefined sets structured as shown in Table 2. Each subject was asked to read three different sets. One set was established as a standard set to be recorded by every speaker while the other two sets were chosen at random for each speaker. During each session, a subject was guided by an interactive program that displayed the utterances to be spoken, allowing the subject move on to succeeding utterances at their own pace. The 57 subjects recorded comprised a roughly equal mix of genders, with 38 native and 19 non-native English speakers. Subject ages ranged from 20 to 65 years old.

A wide variety of trip conditions, including radio and air conditioning use, low and high speed travel, city and highway driving, dry and wet weather, etc. received coverage in the database. Altogether, around 30 hours of raw speech data were collected.

### Table 1: Recording Channel assignment

<table>
<thead>
<tr>
<th>Channel no</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 16</td>
<td>line array, hyper-cardioid</td>
</tr>
<tr>
<td>17</td>
<td>omni-directional mike</td>
</tr>
<tr>
<td>18-20</td>
<td>accelerometers x-y-z axis</td>
</tr>
<tr>
<td>21</td>
<td>boom mike (headset) B&amp;K</td>
</tr>
<tr>
<td>22</td>
<td>boom mike (headset) Sennheiser</td>
</tr>
<tr>
<td>23-24</td>
<td>dashboard cardioid</td>
</tr>
<tr>
<td>25-26</td>
<td>dashboard omni</td>
</tr>
<tr>
<td>27-28</td>
<td>radio left and right ch's</td>
</tr>
<tr>
<td>31</td>
<td>ASCII from laptop PC</td>
</tr>
<tr>
<td>32</td>
<td>GPS</td>
</tr>
</tbody>
</table>

### 3. BLS Acoustic Echo Canceler (AEC)

The acoustic echo cancellation for handsfree communication in general is a more difficult problem than line echo cancellation in telecommunication due to (1) highly variable, possibly long echo return path response; and (2) variable and generally lower SNR’s. The actual time-span of an echo response can, depending upon the physical size of the enclosure, the wall reflectivity, and the relative positioning between speakers and microphones, easily last up to a second. At a sampling rate of 8 kHz, such a room response requires an adaptive filter of several thousand taps. The high computation and memory requirements associated with such a long filter has made the simple, gradient-based Least Mean Squares (LMS) AEC a more desirable solution for commercial applications than its Least Squares (LS) counterpart. However, due to its steepest descent nature, the LMS room echo canceler has certain intrinsic performance disadvantages, including: slower convergence and higher echo residuals. We proposed previously [9] [10] a Block Least-Squares (BLS) algorithm to deliver LS performance but with the computational load of a comparable LMS algorithm for canceling room echoes.

The block LS (BLS) echo canceler algorithm:

1. Block $N$ samples of source signal samples, $x(t)$, and observed microphone input samples, $z(t)$.
2. Weight $x(t)$ and $z(t)$ with an appropriate tapering window.
3. Compute the autocorrelation matrix and cross-correlation vector of the current block.
4. Update $R_{xx}$ and $r_{xz}$ with auto and cross-correlation of the current block using a leaky integrator.
5. Find the adaptive echo canceler coefficients by solving the normal equations via efficient Levinson recursion, $\hat{h} = R_{xz}^{-1} r_{xz}$.
6. Compute the estimated echo, $\hat{y}(t)$, and subtract it from the microphone input sample, $z(t)$. Continue at 1 with the next block.

The computational complexities of the BLS AEC is actually less than a corresponding LMS-based algorithm since the autocorrelation, crosscorrelation estimation and Levinson recursion only need to be done once on a per block, rather than per sample basis.

### AEC Experimental Results

To illustrate the high performance of the new echo canceler and near-end speech detector, both the microphone input and the echo cancelled output around a near-end speech segment are displayed in Figure 2 in both narrowband spectrograms and waveforms. Before BLS cancellation, the far-end echo, due to a long echo response, smears the corresponding spectrogram. When this echo mixes with the near-end speech, near-end speech detection becomes very difficult. After cancellation, while almost all far-end echoes disappear, the near-end speech remains intact and becomes distinctively prominent against the low background noise level. The cancelled output contains almost no audible far-end speech, near-end speech detection becomes very clear, including the weak fricatives at both ends.

### 4. Intelligent Microphone Arrangement

As shown in Figure 1, a hyper-cardioid microphone, due to its directivity pattern, can improve the SNR over an omni-microphone. To further improve the directivity pattern, we can use the delay-and-sum beam-forming technique shown in [3] to adaptively adjust the delay between each microphone element in the array.
such that speech signals from all channels are synchronized for in phase superposition, while the noise sources, being located at various directions from the speech source, are out of phase. When all channels are added together, the speech signals are enhanced and the noise signals are suppressed.

In order to adjust the delay to make an in-phase, delay-and-sum beamforming, we choose one element as a reference and estimate the relative delay between the reference channel and all other array elements. Several time delay estimation techniques have been proposed, such as the generalized correlation method [4] or the eigen analysis based methods [1]. Issues which deserve special attention in estimating the time delays are the sampling rate of each channel and detecting any malfunctional channel in the array. Since the array elements are spaced closely and fractional sample delay will be quantized to its nearest integer sample, the resolution of the delay-and-sum beamforming is reduced. In order to solve it, we can increase the sampling rate. Any malfunctioning channel needs to be detected, otherwise the time delay estimate is misleading and can deteriorate the beamforming performance.

Using the multi-channel signals sampled at 24 KHz with delay-and-sum beam-forming, we managed to improve both SNR' and speech recognition performance [5]. A typical example of waveforms and corresponding spectrograms are shown in Figure 3 before and after the delay-and-sum beamforming. Appreciable noise suppression is observed on the beam-formed output both in the time and frequency domains. Three SNR related measures: segmental SNR, peak-to-background SNR and background noise level are used to calibrate the SNR enhancement performance of the delay-and-sum beamformer over the whole database. As shown in Figure 4, the most significant improvement is from single channel to two channel case, the improvement gradually saturates at an 8-element array. There is no SNR improvement from 8 to 16. This may due to the high correlation between adjacent channels. An automatic speech recognition experiment confirms that the SNR improvement using the delay-and-sum output is useful in reducing the digit string error rate. The string error rate shown in Figure 5 follows the same trend for both mismatched (where a telephone database was used to train the digit models) and matched (where the car data was used to train the model) training and testing conditions.

5. PC Audio Control in a VUI Environment

PC audio I/O handling is a critical aspect of any VUI system. While small time-sharing induced latencies in the ASR, TTS or Database Access modules will generally go unnoticed by users, inability to service audio transfers within strict time constraints will lead to overflow or underrun conditions that result in disturbing interruptions in system prompt output, poor echo cancellation performance, and missed or ignored user speech input.

The conventional approach to audio handling in a non-realtime system is to increase audio buffer sizes, allowing longer grace periods before audio I/O must be serviced. However, in a VUI system with barge-in, where we wish to mute system prompt output, poor echo cancellation performance, and missed or ignored user speech input.
put as soon as we are confident that a user has made a speech utterance, such buffering gives an unwanted, "sluggish" feel to the system since, no matter how quickly we are able to stop writing prompt audio to the output port, the previously buffered audio must drain from the buffer before the prompt actually stops.

Fortunately, Linux audio I/O using the Open Sound System (OSS) [6] enables a unique solution that allows us to meet the contradictory requirements of being immune to small time-sharing latencies and providing nearly immediate system prompt muting.

OSS allows programs to map the soundcard's audio buffers directly into user memory. In our application, 64KB of audio input and output buffers (2 seconds worth of 8kHz, 16 bit stereo audio) are mapped into the Audio Server program's address space. Once started, audio I/O runs continuously to and from these buffers without further program intervention. The audio transfer pointers are circular, continuing at the beginning of the buffer after hitting the end.

The Audio Server blocks (waits) in a SELECT call until the next fragment (typ. 4KB) of the buffer has been transferred. For output, typically a full 2 seconds of audio is waiting in the buffer to be played. When the select returns (indicating that at least one fragment has been played), we check how much has actually been played (using a GETOPTR IOCTL), top off the buffer and block again. In this way we can tolerate up to two seconds of scheduling latency before an underflow condition occurs.

For input, the situation is reversed, and typically the input buffer remains nearly empty. We block waiting for a fragment of input to become available, check to see how much has actually arrived (GETIPTR IOCTL), give it to the echo canceller, and block waiting for more input. In this way we can tolerate up to 2 seconds of service latency before an overflow occurs.

The advantage we get by using memory mapped I/O occurs when the prompt must be muted after barge-in. In this case we ask the system (GETOPTR IOCTL) precisely where the soundcard's output DMA pointer is and then write zeros into the buffer memory following the pointer. This squelches output within a fraction of a second and gives the system a "snappy" barge-in response.

For further refinement, instead of simply zeroing the prompt buffer to mute it, we read the sample values following the output pointer and taper them to zero over a 100ms interval. This gives the pleasing sense that the system is giving up on its utterance by trailing off, much as a human does when interrupted.

A second benefit of the memory mapped, continuously running audio I/O approach occurs when extreme system latencies occasionally do cause buffer over or underruns. In the output buffer underrun case, the circular DMA pointer will play a previous (2 second old) segment of audio. In the input overflow case, new input will overwrite older samples, causing the new audio to appear twice in the input to downstream modules. However despite these discontinuities, the critical relationship between the input and output pointers remains intact. When the Audio Server does catch up and recover, the estimated impulse response has precisely the same flat delay as before. This, combined with the leaky integrator in the Block Least Squares echo canceller, allows echo cancellation performance to remain essentially unaffected even after extreme latencies.

6. REFERENCES