Speech Recognition over Bluetooth Wireless Channels

Ziad Al Bawab, Ivo Locher, Jianxia Xue, and Abeer Alwan

Electrical Engineering Department
University of California, Los Angeles
{zalbawab,ilocher,jxue,alwan}@icsl.ucla.edu

Abstract

This paper studies the effect of Bluetooth wireless channels on distributed speech recognition. An approach for implementing speech recognition over Bluetooth is described. We simulate a Bluetooth environment and then incorporate its performance, in the form of packet loss ratio, into the speech recognition system. We show how intelligent framing of speech feature vectors, extracted by a fixed-point arithmetic front-end, together with an interpolation technique for lost vectors, can lead to a 50.48% relative improvement in recognition accuracy. This is achieved at a distance of 10 meters, around the maximum operating distance between a Bluetooth transmitter and a Bluetooth receiver.

1. Introduction

Smart wireless environments (SWEs) are becoming more popular with today's advancements in research and development of products like wireless sensors, digital signal processors (DSPs), and power supplies (e.g. microbatteries) and with the increasing deployment of wireless area networks (WANs) in businesses, campuses, and residences. One of the useful applications to run in such environments is speech recognition.

In this paper, the SWE studied is composed of voice enabled Bluetooth transmitters (VEBTs) transmitting speech feature vectors to a location server where speech recognition is performed. The application scenario is in a classroom where distributed speech recognition (DSR) is used to monitor the children's interactions by their parents and teachers.

Bluetooth [1] is a low power, low cost, packet-based wireless technology that is frequently used in mobile hand held devices. To our knowledge, no previous study has examined the effect of Bluetooth on DSR. There has been some recent work on the effect of packet loss on DSR [2], [3], but not in the context of Bluetooth applications.

Our goal is to estimate and improve the overall performance of the DSR system in Fig. 1 given the Bluetooth channel. First, we simulate a Bluetooth connection and study its performance focusing on the distance between the transmitter and the receiver. Second, we incorporate the simulation results, the packet loss ratios, into the speech recognition system. We describe an approach for implementing speech recognition over Bluetooth and attempt to improve the system performance by using different framing techniques and by interpolating for lost vectors. In order to closely resemble the operation of the VEBT, we use in our experiments the same feature extraction code, in fixed-point arithmetic, that would be entered into the DSP of the VEBT. Section 2 provides an overview of Bluetooth and its performance effects. Section 3 provides an overview of DSR and describes an approach for implementing it over Bluetooth. Finally, Section 4 discusses the results of the experiments.

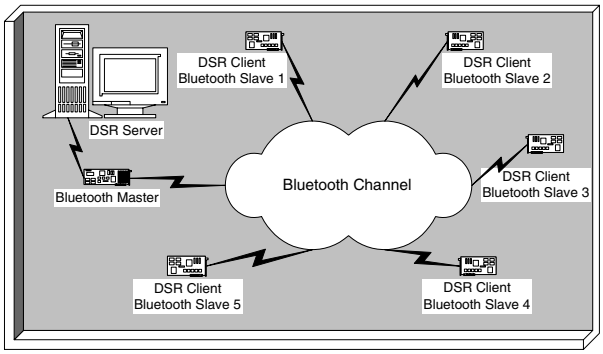

Figure 1: *System overview.*

2. Bluetooth Overview

Bluetooth wireless technology provides ubiquitous solutions interconnecting different devices like laptop computers with mobile phones. Due to the limited power of the Bluetooth antenna (1 mW or 0 dBm), the operating distance is limited to 0–10 meters. A typical Bluetooth connection is composed of a master and 1 up to a maximum of 7 active slaves; which forms a Bluetooth piconet. A master initiates the connection and controls it. Two types of connections exist, synchronous connection oriented (SCO), used for 64 kbps coded speech, and asynchronous connectionless link (ACL), used for other kinds of data with a maximum rate of 723.2 kbps [1]. In this section we focus on what affects Bluetooth's reliability. We consider loss due to path and interference. We also mention the techniques applied to correct packet loss and discuss different packet types.

2.1. Path Loss

In wireless communications, path loss is mainly due to the distance d between the transmitter and the receiver and to shadowing and fading acting on the channel between them. In general, efficiency decreases with distance due to the limited transmission power. More bit errors happen causing frame droppings at the receiver when error concealment techniques fail. The shadowing factor, $s(d)$, can be modelled as a log-normally distributed random variable with a zero mean and a variance function of d, $10 \lg s(d) = \mathbf{N}(0, \sigma)$ [4]. The fading factor can also be modelled as an exponential random variable with a unity mean.

2.2. Interference

The Bluetooth standard uses frequency hopping technique to select the frequency to which members of a piconet are tuned each time a transmission takes place. It selects from 79 available

channels, each of width 1 MHz, spread around 2.45 GHz, and falling in the industrial, scientific, and medical (ISM) free band (2.4–2.483 GHz). Bluetooth uses time division duplex (TDD) where each device is given the chance to use the channel. This prevents 2 or more members from transmitting at the same time and in turn prevents crosstalk from within the piconet. Interference from other piconets is possible if transmission overlaps in time and frequency [4].

2.3. Error Correction and Packet Types

Bluetooth deploys 2 techniques to correct errors, Forward Error Correction (FEC) and Automatic Repeat Request (ARQ). With a 2/3 rate FEC, a (15–10) shortened Hamming code is applied to the data payload and the resulting parity bits are transmitted with the payload. 2/3 FEC can correct a 1-bit error and can detect a 2-bit error in every 10 bits. With ARQ, the transmitter retransmits the same packet until either a positive acknowledgement (ACK) is received or the number of retransmissions exceeds a certain threshold which depends on the time-sensitivity of the data. Both techniques cause additional overhead in transmission, especially in error free connections (by FEC), or in bursty channels (by ARQ).

Bluetooth data packets are composed of an access code, a packet header, and a data payload. Data packets are classified according to connection type (ACL or SCO), FEC support, and the number of time slots occupied by the packet. The smallest time unit or slot is 625 μs. 1, 3, or 5 time-slot packets are specified. Payload size varies accordingly. For example, DM5 packets are used in ACL connections. They use 2/3 FEC and occupy 5 time slots with a maximum payload size of 224 bytes (121 bytes for DM3). It is expected that longer packets experience more path loss and interference effects than shorter packets.

3. Distributed Speech Recognition Overview

We adopt a DSR approach in order to add speech recognition as an application to our SWE. Recent research concerned with recognizing speech over networks has proven DSR to be the prominent choice for such applications [5]. We follow the ETSI STQ-Aurora DSR standard [6] for the front-end feature extraction, feature vector quantization, and framing specifications. In this section we describe these steps and their implementation in fixed-point arithmetic. We also describe the framing techniques followed to fit the quantized feature vectors into multi-frames to be carried by Bluetooth packets. Three different framing and Bluetooth packetization approaches are explained. We end this section with a description of the interpolation technique for lost feature vectors estimation.

3.1. Front-end Feature Extraction and Vector Quantization

We process speech according to the ETSI STQ-Aurora DSR standard. The DC components of the speech signal are removed using a notch filter. Framing divides the speech sample stream into frames of 25 ms duration (400 samples at 16kHz sampling frequency) with a frame shift of 10 ms (160 samples). Further, we calculate logE, the natural logarithm of the frame energy. Pre-emphasis is used to boost up the higher frequency components in the spectrum. A Hamming window of 400 samples width is then applied to the speech frames. The 13 Mel-frequency cepstral coefficients are then computed with discrete cosine transform (DCT). The signal energy is compacted and shifted down to c_0, the 0^{th} cepstral coefficient. This coefficient is comparable with logE. The Aurora standard defines a 14 dimensional feature vector as $[c_1, c_2, \ldots, c_{12}, c_0, logE]$.

In addition, Aurora applies vector quantization to the feature vector. The codebook used is fixed to $64\,(2^6)$ entries for each of the 6 cepstral pairs (c_1, c_2), (c_3, c_4), $\ldots$,(c_{11}, c_{12}) and $256\,(2^8)$ entries for the $(c_0, logE)$ pair. Hence, 6 bits are needed for each cepstral pair and 8 bits for the $(c_0, logE)$ pair. Therefore, a speech feature vector is quantized into a 44-bit frame.

3.2. Fixed-Point Arithmetic Implementation of the Front-end

In order to mimic the behavior of our system and closely estimate the performance of DSR over Bluetooth, we translate the existing floating-point front-end code into fixed-point code that would be entered into the DSP of the VEBT. The floating-point code is based on Nokia's code, which is provided on ETSI's web-site (www.etsi.org).

The code requires large restructuring in order to introduce real-time streaming capability. Due to fixed-point translation, execution time decreases dramatically such that the limited processing power of a fixed-point DSP would be able to satisfy real-time performance. A big problem in fixed-point operations lies in the much smaller dynamic range compared to floating-point operations. Fractions require scaling-up and rounding to integers such that the targeted accuracy can be achieved. Front-end processing uses transformations that allow table look-ups instead of on the fly calculations. Thus, many operations are adapted in this way with proper scaling in order to speed up the computation. Using these methods requires a higher amount of static memory resources. Table 1 gives an overview of the look-up tables applied in the code. Note that since Hamming window is symmetric, only half of it has to be stored. Since sine and cosine are affine, only a quarter of a sine has to be stored. As for the Mel-filter bank, there are different numbers of coefficients for each filter. For the cosine matrix, 12 cepstral coefficients are required without the 0^{th} entry since the cosine is then always one. There are 23 entries for each coefficient in the DCT calculation. Further, the generic functions "square root" and "natural logarithm" are implemented in fixed-point whereby different algorithms such as those in [7] and [8] were tested regarding best performances.

Table 1: *Look-up tables for front-end processing.*

Operation	Table for	Table Size
Windowing	Window Coefficients	200
FFT (512pt)	Sine, Cosine	128
Mel-Filtering	Filter Bank	up to 58
DCT	Cosine Matrix	276

In contrast to the floating-point version, feature extraction and vector quantization are combined in the fixed-point version in order to avoid loss of accuracy. The complete fixed-point code is available at http://nesl.ee.ucla.edu/projects/ibadge/software.htm.

3.3. Framing and Bluetooth Packetization

As mentioned in subsection 3.1, each feature vector is quantized into a 44-bit frame. Framing, as specified by the Aurora standard, adds a 4-bit CRC for each frame pair, hence a total of 92 bits for each pair. The final bit stream framing format collects

24 frames into a multi-frame with a synchronization sequence (16 bits) and a header (32 bits) before sending them to the transport protocol. The multi-frame formation time is 240 ms (frame rate is 10 ms) and it consists of 144 bytes. This makes up a bit rate or throughput of 4.8 kbps. As mentioned in subsection 2.3, a DM5 Bluetooth packet has a maximum payload size of 224 bytes, which can clearly handle a multi-frame. We refer to this framing technique as technique 1.

Another framing technique, technique 2, that we use is to modify the Aurora standard such that a multi-frame contains 20 frames. In this case the formation time is 200 ms and it consists of 121 bytes that fit exactly into a DM3 packet maximum payload, at a throughput of 4.84 kpbs. In this case, the packet length is shorter, which means more robust to interferences, yet a higher packet overhead of 40.69% compared to 39.75% in the previous case. Overhead is due to access code, header, and payload header and CRC.

We also use a third framing technique, technique 3, that boosts up the bit rate making use of the available Bluetooth bandwidth. We modify the multi-frame to contain 18 frames from newly extracted and quantized vectors and 18 frames repeated from the previous multi-frame. Hence, the multi-frame formation time is 180 ms and it consists of 213 bytes fitting into a DM5 packet, at a throughout of 9.47 kbps. The overhead in the Bluetooth packet is also reduced to 37.72%. In spite of the redundancy added to the payload, the same amount of data is sent since the bit rate is higher in this case. Table 2 summarizes these 3 techniques.

Table 2: *Framing techniques specifications.*

Technique	Number of Frames	Multi-Frame Size (Bytes)	Bluetooth Packet Type, Total Size
1	24 (all new)	144	DM5, 239
2	20 (all new)	121	DM3, 204
3	36 (18 new)	213	DM5, 342

3.4. Interpolation for Lost Feature Vectors

Interpolation is one of the solutions to the problem of feature vector loss as proposed in several papers (e.g. [2]). Since loss actually happens for Bluetooth packets carrying multi-frames of 24, 20, or 36 feature vectors (or frames), then interpolation should be applied for at least this number of vectors lost (18 in the case of technique 3 when 2 consecutive multi-frames are lost).

We interpolate for lost feature vectors using the last vector received before the loss starts $V(lstart-1)$ and the first vector received after the loss ends $V(lend+1)$. We use a procedure similar to that of the GSM standard for the substitution and muting of lost frames [9]. To estimate the lost vector, $V'(i)$, we apply a trigonometric weight (1) to $V(lstart-1)$ in the first half of the loss region (2) until it is zeroed in the middle of the region. Similarly we continue applying the weight function to $V(lend+1)$ in the second half of the loss region (3). We account for loss happening at the beginning or end of transmission in a similar fashion.

$$W(i) = \frac{1+\cos 2\pi \frac{i-lstart}{lend-lstart}}{2} \quad (1)$$

$$V'(i) = V(lstart-1) * W(i), \quad i = lstart : \frac{lstart+lend}{2} \quad (2)$$

$$V'(i) = V(lend+1) * W(i), \quad i = \frac{lstart+lend}{2}+1 : lend \quad (3)$$

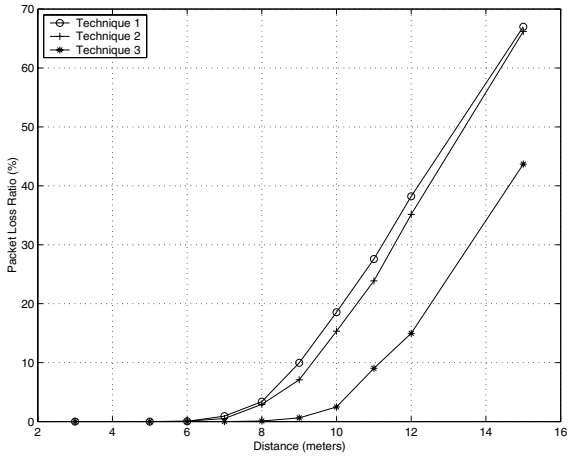

Figure 2: *Bluetooth performance results for the 3 framing techniques.*

4. Experimental Results

Our experiments require two steps. In the first step we simulate a Bluetooth connection. In the second we incorporate the Bluetooth simulation results into the DSR system and run the DSR experiments.

4.1. Bluetooth Simulations

We use BlueHoc [10], IBM's open-source Bluetooth simulator, which is an extension of ns (Network Simulator) [11]. BlueHoc's model for Bluetooth channels and frame error rates (FER) are derived and calculated in [12]. FERs are computed for different Bluetooth packet types taking into consideration the distance effect and other path loss effects discussed in subsection 2.1.

We run 60 secs point-to-point ACL experiments between a master and a slave. UDP is chosen as the transport protocol for the traffic data. The size of the UDP dummy packets is adjusted according to the 3 multi-frame sizes discussed in subsection 3.3. UDP does not require retransmissions as TCP. Since DM3 and DM5 packets use ARQ, then retransmission is already implemented in the lower Bluetooth link layers. Yet, we limit the retransmission threshold to 1 since our data is time-sensitive. Only in the case of framing technique 3 that DSR data (18 feature vectors) loss is announced when 2 consecutive multi-frames (or packets) are lost due to the redundancy between them. In the other techniques, a packet loss is equivalent to a multi-frame loss. For convenience, we define packet loss ratio differently for the case of technique 3.

In each simulation, we record the ratio of packets lost. The results are averaged over 10 simulations for each of the 10 distances we study. Fig. 2 displays the results. As mentioned in Section 2, Bluetooth operating distance is limited to 10 meters. Beyond this distance, packet loss ratio increases significantly. With technique 2, the performance is better than with technique 1 since the Bluetooth packets used are shorter. Technique 3 yields better results than the rest due to redundancy. This will reflect on DSR performance as will be shown below.

4.2. DSR Experiments

In DSR, training and testing are performed on the server side. The first step would be to train the HMMs on a set of feature

vectors extracted from the training utterances and then quantized. On the client side, the feature vectors are extracted from the speech recorded by the VEBT. Feature vectors are then quantized, packetized, and transported by Bluetooth to the server using the techniques explained in subsection 3.3. Refer to Fig. 1 for a system overview.

HTK 3.1 is used for training and testing with HMMs of 16 states and 3 Gaussian mixtures per state. We use connected digits speech utterances from the TIDIGITS database resampled to 16 kHz. The database contains training and testing samples of 4 subjects: a young boy, a young girl, an adult man, and an adult woman, with 6325, 6312, 14159, and 14459 words to test for each subject respectively. Hereafter, we refer to these subjects as "Boy", "Girl", "Man", and "Woman" respectively. A similar number of words is used in training for each subject separately.

Feature extraction and quantization on the client and the server sides use the front-end explained in subsections 3.1 and 3.2. For training, we calculate the 1^{st} and 2^{nd} order derivatives of the feature vectors such that we train on 42 dimensional vectors. To test the HMMs, we first apply the Bluetooth performance results to the multi-frames in order to simulate the Bluetooth channel effect. We zero the lost vectors instead of dropping them, hence preserving the time alignment information. The lost feature vectors are then estimated using the technique of subsection 3.4. Finally, we calculate the 1^{st} and 2^{nd} order derivatives and test on 42 dimensional feature vectors.

For each subject, we train the HMMs with no packet loss effects. We then test them for each of the 10 Bluetooth distances. This is repeated for each of the 3 framing techniques. For the "Girl" subject, Fig. 3 shows the recognition results. Notice how the 3 framing techniques give different recognition accuracies, whereby the best performance is reached by technique 3. The larger the distance the poorer the performance of DSR. Interpolation helps in improving the accuracies especially at larger distances with increasing packet loss ratios. For example, at a distance of 10 meters, technique 1 gives a recognition accuracy of 63.05% (77.63% with interpolation), whereas technique 3 gives 94.88% with interpolation. Hence the combination of framing technique 3 and interpolation leads to a 50.48% relative improvement in recognition accuracy. Table 3 shows the results for the 4 subjects using technique 3 with interpolation. Results vary amongst subjects due to differences in gender, age, and the amount of training and testing data used.

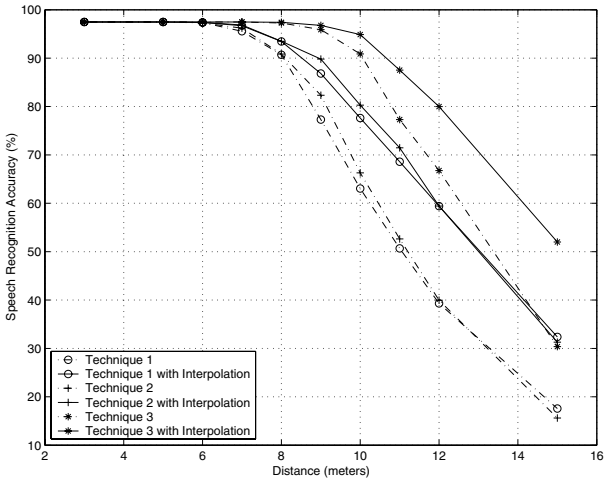

Figure 3: *DSR performance for the 3 framing techniques showing the improvement with interpolation. The subject is the "Girl" and the database is TIDIGITS.*

Table 3: *DSR performance for the 4 subjects using technique 3 with interpolation.*

Distance	Packet Loss Ratio (%)	Boy	Girl	Man	Woman	Average
3	0.00	96.1	97.5	97.4	98.3	97.3
5	0.00	96.1	97.5	97.4	98.3	97.3
6	0.00	96.1	97.5	97.4	98.3	97.3
7	0.00	96.1	97.5	97.4	98.3	97.3
8	0.09	96.0	97.5	97.3	98.3	97.3
9	0.63	95.3	96.8	96.7	97.6	96.6
10	2.47	93.0	94.9	94.9	95.6	94.6
11	9.03	84.8	87.5	88.4	88.2	87.2
12	14.97	78.6	80.0	82.8	81.0	80.6
15	43.69	50.2	52.0	54.7	52.9	52.5

5. Summary

In this paper, we have described an approach for implementing speech recognition over Bluetooth. Using computer simulations, we have shown the effect of Bluetooth operating distances on recognition accuracies. We conclude that intelligent framing and interpolation techniques can lead to improving recognition performance in DSR environments. This study can be helpful for DSR applications over other wireless or even wired networking protocols. This work was supported in part by the NSF.

6. References

[1] J. C. Haartsen, "The Bluetooth Radio System," *IEEE Personal Comm*, vol. 7, pp. 28-36, February 2000.

[2] B. Milner, "Robust speech recognition in burst-like packet loss," in *Proc. of ICASSP*, vol. 1, pp. 261-264, May 2001.

[3] A. Bernard and A. Alwan, "Low-bitrate distributed speech recognition for packet-based and wireless communication," *IEEE Trans. on Speech and Audio Proc*, Vol. 10, Number 8, pp. 570-580, Nov. 2002.

[4] S. Zurbes, W. Stahl, K. Matheus, and J. Haarsten, "Radio network performance of Bluetooth," in *Proc. of IEEE ICC 2000*, vol. 3, pp. 1563-1567, New Orleans, LA, June 2000.

[5] V. Digalakis, L. Neumeyer, and M. Perakakis, "Quantization of cepstral parameters for speech recognition over the World Wide Web," *IEEE J-SAC*, vol. 17, no. 1, pp. 82-90, January 1999.

[6] "ETSI ES 201 108 v1.1.2 Distributed Speech Recognition; Front-end Feature Extraction Algorithm; Compression Algorithm," April 2000.

[7] J. W. Crenshaw, Math Toolkit for Real-Time Programming, ISBN 1-929629-09-5.

[8] R. Andraka, A survey of CORDIC algorithms for FPGA based computers, Andraka Consulting Group, North Kingstown.

[9] "ETSI En 300 970 v8.0.1 Digital cellular telecommunications system (Phase 2+); Half rate speech; Substitution and muting of lost frames for half rate speech traffic channels (GSM 06.21 version 8.0.1 Release 1999)," November 2000.

[10] BlueHoc: Open-source Bluetooth Simulator (extension of ns-2.1b8a) available from IBM developerWorks: http://oss.software.ibm.com/developerworks/projects/bluehoc

[11] ns (Network Simulator) manual: http://www.isi.edu/nsnam/ns/ns-documentation.html

[12] A. Kumar and R. Gupta, "Capacity evaluation of frequency hopping based ad-hoc systems," in *ACM Sigmetrics'2001*, Boston, MA, June, 2001.