Probabilistic Integration of Partial Lexical Information for Noise Robust Haptic Voice Recognition

Khe Chai Sim
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417
simkc@comp.nus.edu.sg

Abstract

This paper presents a probabilistic framework that combines multiple knowledge sources for Haptic Voice Recognition (HVR), a multimodal input method designed to provide efficient text entry on modern mobile devices. HVR extends the conventional voice input by allowing users to provide complementary partial lexical information via touch input to improve the efficiency and accuracy of voice recognition. This paper investigates the use of the initial letter of the words in the utterance as the partial lexical information. In addition to the acoustic and language models used in automatic speech recognition systems, HVR uses the haptic and partial lexical models as additional knowledge sources to reduce the recognition search space and suppress confusions. Experimental results show that both the word error rate and runtime factor can be reduced by a factor of two using HVR.

1 Introduction

Nowadays, modern portable devices, such as the smartphones and tablets, are equipped with microphone and touchscreen display. With these devices becoming increasingly popular, there is an urgent need for an efficient and reliable text entry method on these small devices. Currently, text entry using an onscreen virtual keyboard is the most widely adopted input method on these modern mobile devices. Unfortunately, typing with a small virtual keyboard can sometimes be cumbersome and frustratingly slow for many people. Instead of using a virtual keyboard, it is also possible to use handwriting gestures to input text. Handwriting input offers a more convenient input method for writing systems with complex orthography, including many Asian languages such as Chinese, Japanese and Korean. However, handwriting input is not necessarily more efficient compared to keyboard input for English. Moreover, handwriting recognition is susceptible to recognition errors, too.

Voice input offers a hands-free solution for text entry. This is an attractive alternative for text entry because it completely eliminates the need for typing. Voice input is also more natural and faster for human to convey messages. Normally, the average human speaking rate is approximately 100 words per minute (WPM). Clarkson et al. (2005) showed that the typing speed for regular users reaches only 86.79 – 98.31 using a full-size keyboard and 58.61 – 61.44 WPM using a mini-QWERTY keyboard. Evidently, speech input is the preferred text entry method, provided that speech signals can be reliably and efficiently converted into texts. Unfortunately, voice input relies on automatic speech recognition (ASR) (Rabiner, 1989) technology, which requires high computational resources and is susceptible to performance degradation due to acoustic interference, such as the presence of noise.

In order to improve the reliability and efficiency of ASR, Haptic Voice Recognition (HVR) was proposed by Sim (2010) as a novel multimodal input method combining both speech and touch inputs. Touch inputs are used to generate *haptic events*, which correspond to the initial letters of the words in the spoken utterance. In addition to the regular beam

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 31–39,
Jeju, Republic of Korea, 8-14 July 2012.

pruning used in traditional ASR (Ortmanns et al., 1997), search paths which are inconsistent with the haptic events are also pruned away to achieve further reduction in the recognition search space. As a result, the runtime of HVR is generally more efficient than ASR. Furthermore, haptic events are not susceptible to acoustic distortion, making HVR more robust to noise.

This paper proposes a probabilistic framework that encompasses multiple knowledge sources for combining the speech and touch inputs. This framework allows coherent probabilistic models of different knowledge sources to be tightly integrated. In addition to the acoustic model and language model used in ASR, *haptic model* and *partial lexical model* are also introduced to facilitate the integration of more sophisticated haptic events, such as the keystrokes, into HVR.

The remaining of this paper is organised as follows. Section 2 gives an overview of existing techniques in the literature that aim at improving noise robustness for automatic speech recognition. Section 3 gives a brief introduction to HVR. Section 4 proposes a probabilistic framework for HVR that unifies multiple knowledge sources as an integrated probabilistic generative model. Next, Section 5 describes how multiple knowledge sources can be integrated using Weighted Finite State Transducer (WFST) operations. Experimental results are presented in Section 6. Finally, conclusions are given in Section 7.

2 Noise Robust ASR

As previously mentioned, the process of converting speech into text using ASR is error-prone, where significant performance degradation is often due to the presence of noise or other acoustic interference. Therefore, it is crucial to improve the robustness of voice input in noisy environment. There are many techniques reported in the literature which aim at improving the robustness of ASR in noisy environment. These techniques can be largely divided into two groups: 1) using speech enhancement techniques to increase the signal-to-noise ratio of the noisy speech (Ortega-Garcia and Gonzalez-Rodriguez, 1996); and 2) using model-based compensation schemes to *adapt* the acoustic models to noisy environment (Gales and Young, 1996; Acero et al., 2000).

From the information-theoretic point of view, in order to achieve reliable information transmission, *redundancies* are introduced so that information lost due to channel distortion or noise corruption can be recovered. Similar concept can also be applied to improve the robustness of voice input in noisy environment. Additional complementary information can be provided using other input modalities to provide *cues* (redundancies) to boost the recognition performance. The next section will introduce a multimodal interface that combines speech and touch inputs to improve the efficiency and noise robustness for text entry using a technique known as Haptic Voice Recognition (Sim, 2010).

3 Haptic Voice Recognition (HVR)

For many voice-enabled applications, users often find voice input to be a *black box* that captures the users' voice and automatically converts it into texts using ASR. It does not provide much flexibility for human intervention through other modalities in case of errors. Certain applications may return multiple hypotheses, from which users can choose the most appropriate output. Any remaining errors are typically corrected manually. However, it may be more useful to give users more control during the input stage, instead of having a post-processing step for error correction. This motivates the investigation of multimodal interface that tightly integrates speech input with other modalities.

Haptic Voice Recognition (HVR) is a multimodal interface designed to offer users the opportunity to add his or her '*magic touch*' in order to improve the accuracy, efficiency and robustness of voice input. HVR is designed for modern mobile devices equipped with an embedded microphone to capture speech signals and a touchscreen display to receive touch events. The HVR interface aims to combine both speech and touch modalities to enhance speech recognition. When using an HVR interface, users will input text verbally, at the same time provide additional cues in the form of *Partial Lexical Information* (PLI) to guide the recognition search. PLIs are simplified lexical representation of words that should be easy to enter whilst speaking (*e.g.* the

prefix and/or suffix letters). Preliminary simulated experiments conducted by Sim (2010) show that potential performance improvements both in terms of recognition speed and noise robustness can be achieved using the initial letters as PLIs. For example, to enter the text "*Henry will be in Boston next Friday*", the user will speak the sentence and enter the following letter sequence: 'H', 'W', 'B', 'I', 'B', 'N' and 'F'. These additional letter sequence is simple enough to be entered whilst speaking; and yet they provide crucial information that can significantly improve the efficiency and robustness of speech recognition. For instance, the number of letters entered can be used to constrain the number of words in the recognition output, thereby suppressing spurious insertion and deletion errors, which are commonly observed in noisy environment. Furthermore, the identity of the letters themselves can be used to guide the search process so that partial word sequences in the search graph that do not conform to the PLIs provided by the users can be pruned away.

PLI provides additional complementary information that can be used to eliminate confusions caused by poor speech signal. In conventional ASR, acoustically similar word sequences are typically resolved implicitly using a language model where contexts of neighboring words are used for disambiguation. On the other hand, PLI can also be very effective in disambiguating homophones[1] and similar sounding words and phrases that have distinct initial letters. For example, '*hour*' versus '*our*', '*vary*' versus '*marry*' and '*great wine*' versus '*grey twine*'.

This paper considers two methods of generating the initial letter sequence using a touchscreen. The first method requires the user to tap on the appropriate keys on an onscreen virtual keyboard to generate the desired letter sequence. This method is similar to that proposed in Sim (2010). However, typing on small devices like smartphones may require a great deal of concentration and precision from the users. Alternatively, the initial letters can be entered using handwriting gestures. A gesture recognizer can be used to determine the letters entered by the users. In order to achieve high recognition accuracy, each letter is represented by a single-stroke gesture, so that isolated letter recognition can be performed. Figure 1 shows the single-stroke gestures that are used in this work.

[1]Words with the same pronunciation

4 A Probabilistic Formulation for HVR

Let $\mathcal{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$ denote a sequence of T observed acoustic features such as MFCC (Davis and Mermelstein, 1980) or PLP (Hermansky, 1990) and $\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N\}$ denote a sequence of N haptic features. For the case of keyboard input, each $\boldsymbol{h}_i$ is a discrete symbol representing one of the 26 letters. On the other hand, for handwriting input, each $\boldsymbol{h}_i$ represents a sequence of 2-dimensional vectors that corresponds to the coordinates of the points of the keystroke. Therefore, the haptic voice recognition problem can be defined as finding the *joint* optimal solution for both the word sequence, $\hat{\mathcal{W}}$ and the PLI sequence, $\hat{\mathcal{L}}$, given $\mathcal{O}$ and $\mathcal{H}$. This can be expressed using the following formulation:

$$(\hat{\mathcal{W}}, \hat{\mathcal{L}}) = \arg\max_{\mathcal{W},\mathcal{L}} P(\mathcal{W}, \mathcal{L}|\mathcal{O}, \mathcal{H}) \quad (1)$$

where according to the Bayes' theorem:

$$\begin{aligned} P(\mathcal{W}, \mathcal{L}|\mathcal{O}, \mathcal{H}) &= \frac{p(\mathcal{O}, \mathcal{H}|\mathcal{W}, \mathcal{L})P(\mathcal{W}, \mathcal{L})}{p(\mathcal{O}, \mathcal{H})} \\ &= \frac{p(\mathcal{O}|\mathcal{W})p(\mathcal{H}|\mathcal{L})P(\mathcal{W}, \mathcal{L})}{p(\mathcal{O}, \mathcal{H})} \end{aligned} \quad (2)$$

The joint prior probability of the observed inputs, $p(\mathcal{O}, \mathcal{H})$, can be discarded during the maximisation of Eq. 1 since it is independent of $\mathcal{W}$ and $\mathcal{L}$. $p(\mathcal{O}|\mathcal{W})$ is the acoustic likelihood of the word sequence, $\mathcal{W}$, generating the acoustic feature sequence, $\mathcal{O}$. Similarly, $P(\mathcal{H}|\mathcal{L})$ is the *haptic likelihood* of the lexical sequence, $\mathcal{L}$, generating the observed haptic inputs, $\mathcal{H}$. The *joint* prior probability, $P(\mathcal{W}, \mathcal{L})$, can be decomposed into:

$$P(\mathcal{W}, \mathcal{L}) = P(\mathcal{L}|\mathcal{W})P(\mathcal{W}) \quad (3)$$

where $P(\mathcal{W})$ can be modelled by the word-based n-gram language model (Chen and Goodman, 1996) commonly used in automatic speech recognition. Combining Eq. 2 and Eq. 3 yields:

$$\begin{aligned} &P(\mathcal{W}, \mathcal{L}|\mathcal{O}, \mathcal{H}) \propto \\ &\quad p(\mathcal{O}|\mathcal{W}) \times p(\mathcal{H}|\mathcal{L}) \times P(\mathcal{L}|\mathcal{W}) \times P(\mathcal{W}) \end{aligned} \quad (4)$$

It is evident from the above equation that the probabilistic formulation of HVR combines four knowledge sources:

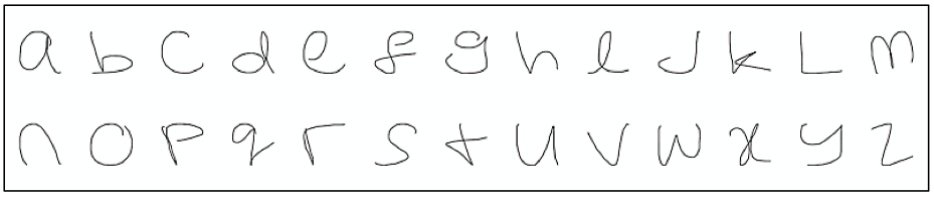

Figure 1: Examples of single-stroke handwriting gestures for the 26 English letters

- Acoustic model score: $p(\mathcal{O}|\mathcal{W})$
- Haptic model score: $p(\mathcal{H}|\mathcal{L})$
- PLI model score: $P(\mathcal{L}|\mathcal{W})$
- Language model score: $P(\mathcal{W})$

Note that the acoustic model and language model scores are already used in the conventional ASR. The probabilistic formulation of HVR incorporated two additional probabilities: *haptic model score*, $p(\mathcal{H}|\mathcal{L})$ and *PLI model score*, $P(\mathcal{L}|\mathcal{W})$. The role of the haptic model and PLI model will be described in the following sub-sections.

4.1 Haptic Model

Similar to having an acoustic model as a statistical representation of the phoneme sequence generating the observed acoustic features, a haptic model is used to model the PLI sequence generating the observed haptic inputs, $\mathcal{H}$. The haptic likelihood can be factorised as

$$p(\mathcal{H}|\mathcal{L}) = \prod_{i=1}^{N} p(\boldsymbol{h}_i|\boldsymbol{l}_i) \tag{5}$$

where $\mathcal{L} = \{\boldsymbol{l}_i : 1 \leq i \leq N\}$. $\boldsymbol{l}_i$ is the ith PLI in $\mathcal{L}$ and $\boldsymbol{h}_i$ is the ith haptic input feature. In this work, each PLI represent the initial letter of a word. Therefore, $\boldsymbol{l}_i$ represents one of the 26 letters. As previously mentioned, for keyboard input, $\boldsymbol{h}_i$ are discrete features whose values are also one of the 26 letters. Therefore, $p(\boldsymbol{h}_i|\boldsymbol{l}_i)$ forms a 26×26 matrix. A simple model can be derived by making $p(\boldsymbol{h}_i|\boldsymbol{l}_i)$ an identity matrix. Therefore, $p(\boldsymbol{h}_i|\boldsymbol{l}_i) = 1$ if $\boldsymbol{h}_i = \boldsymbol{l}_i$; otherwise, $p(\boldsymbol{h}_i|\boldsymbol{l}_i) = 0$. However, it is also possible to have a non-diagonal matrix for $p(\boldsymbol{h}_i|\boldsymbol{l}_i)$ in order to accommodate typing errors, so that non-zero probabilities are assigned to cases where $\boldsymbol{h}_i \neq \boldsymbol{l}_i$. For handwriting input, $\boldsymbol{h}_i$ denote a sequence of 2-dimensional feature vectors, which can be modelled using Hidden Markov Models (HMMs) (Rabiner, 1989). Therefore, $(\boldsymbol{h}_i|\boldsymbol{l}_i)$ is simply given by the HMM likelihood. In this work, each of the 26 letters is represented by a left-to-right HMM with 3 emitting states.

4.2 Partial Lexical Information (PLI) Model

Finally, a PLI model is used to impose the compatibility constraint between the PLI sequence, $\mathcal{L}$, and the word sequence, $\mathcal{W}$. Let $\mathcal{W} = \{\boldsymbol{w}_i : 1 \leq i \leq M\}$ denote a word sequence of length M. If $M = N$, the PLI model likelihood, $P(\mathcal{L}|\mathcal{W})$, can be expressed in the following form:

$$P(\mathcal{L}|\mathcal{W}) = \prod_{i=1}^{N} P(\boldsymbol{l}_i|\boldsymbol{w}_i) \tag{6}$$

where $P(\boldsymbol{l}_i|\boldsymbol{w}_i)$ is the likelihood of the ith word, $\boldsymbol{w}_i$, generating the ith PLI, $\boldsymbol{l}_i$. Since each word is represented by a unique PLI (the initial letter) in this work, the PLI model score is given by

$$P(\boldsymbol{l}_i|\boldsymbol{w}_i) = C_{\texttt{sub}} = \begin{cases} 1 & \text{if } \boldsymbol{l}_i = \text{initial letter of } \boldsymbol{w}_i \\ 0 & \text{otherwise} \end{cases}$$

On the other hand, if $N \neq M$, *insertions* and *deletions* have to be taken into consideration:

$$P(\boldsymbol{l}_i = \epsilon|\boldsymbol{w}_i) = C_{\texttt{del}} \quad \text{and} \quad P(\boldsymbol{l}_i|\boldsymbol{w}_i = \epsilon) = C_{\texttt{ins}}$$

where ϵ represents an empty token. $C_{\texttt{del}}$ and $C_{\texttt{ins}}$ denote the deletion and insertion penalties respectively. This work assumes $C_{\texttt{del}} = C_{\texttt{ins}} = 0$. This means that the word count of the HVR output matches the length of the initial letter sequence entered by the user. Assigning a non-zero value to $C_{\texttt{del}}$ gives the users option to skip entering letters for certain words (*e.g.* short words).

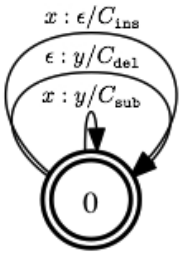

Figure 2: WSFT representation of PLI model, $\bar{\mathcal{P}}$

5 Integration of Knowledge Sources

As previously mentioned, the HVR recognition process involves maximising the posterior probability in Eq. 4, which can be expressed in terms of four knowledge sources. It turns out that these knowledge sources can be represented as Weighted Finite State Transducers (WFSTs) (Mohri et al., 2002) and the *composition* operation ($\circ$) can be used to integrate these knowledge sources into a single WFST:

$$\bar{\mathcal{F}}_{\texttt{integrated}} = \bar{\mathcal{A}} \circ \bar{\mathcal{L}} \circ \bar{\mathcal{P}} \circ \bar{\mathcal{H}} \tag{7}$$

where $\bar{\mathcal{A}}$, $\bar{\mathcal{L}}$, $\bar{\mathcal{P}}$ and $\bar{\mathcal{H}}$ denote the WFST representation of the acoustic model, language model, PLI model and haptic model respectively. Mohri et al. (2002) has shown that Hidden Markov Models (HMMs) and n-gram language models can be viewed as WFSTs. Furthermore, HMM-based haptic models are also used in this work to represent the single-stroke letters shown in Fig. 1. Therefore, $\bar{\mathcal{A}}$, $\bar{\mathcal{L}}$, and $\bar{\mathcal{H}}$ can be obtained from the respective probabilistic models. Finally, the PLI model described in Section 4.2 can also be represented using the WFST as shown in Fig. 2. The transition weights of these WFSTs are given by the negative log probability of the respective models. $\bar{\mathcal{P}}$ can be viewed as a merger that defines the possible alignments between the speech and haptic inputs. Each complete path in $\bar{\mathcal{F}}$ represents a valid pair of $\mathcal{W}$ and $\mathcal{L}$ such that the weight of the path is given by the *negative* $\log P(\mathcal{L}, \mathcal{W} | \mathcal{O}, \mathcal{H})$. Therefore, finding the shortest path in $\bar{\mathcal{F}}$ is equivalent to solving Eq. 1.

Direct decoding from the overall composed WFST, $\bar{\mathcal{F}}_{\texttt{integrated}}$, is referred to as *integrated decoding*. Alternatively, HVR can also operate in a *lattice rescoring* manner. Speech input and haptic input are processed separately by the ASR system and the haptic model respectively. The ASR system may

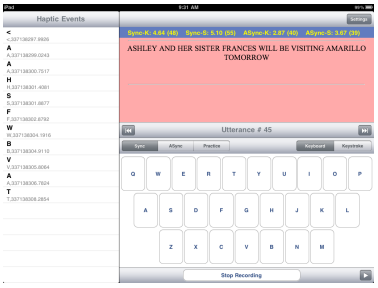

Figure 3: Screenshot depicting the HVR prototype operating with keyboard input

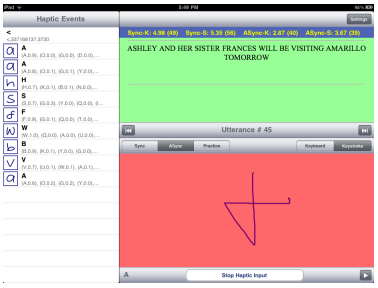

Figure 4: Screenshot depicting the HVR prototype operating with keystroke input

generate multiple hypotheses of word sequences in the form of a lattice. Similarly, the haptic model may also generate a lattice containing the most probably letter sequences. Let $\hat{\mathcal{L}}$ and $\hat{\mathcal{H}}$ represent the word and letter lattices respectively. Then, the final HVR output can be obtained by searching for the shortest path of the following merged WFST:

$$\bar{\mathcal{F}}_{\texttt{rescore}} = \hat{\mathcal{L}} \circ \bar{\mathcal{P}} \circ \hat{\mathcal{H}} \tag{8}$$

Note that the above composition may yield an empty WFST. This may happen if the lattices generated by the ASR system or the haptic model are not large enough to produce any valid pair of $\mathcal{W}$ and $\mathcal{L}$.

6 Experimental Results

In this section, experimental results are reported based on the data collected using a prototype HVR interface implemented on an iPad. This prototype HVR interface allows both speech and haptic input data to be captured either synchronously or asynchronously and the partial lexical information can be entered using either a soft keyboard or handwriting gestures. Figures 3 and 4 shows the screenshot of the HVR prototype iPad app using the key-

Donna was in Cincinnati last Thursday.
Adam will be visiting Charlotte tomorrow
Janice will be in Chattanooga next month.
Christine will be visiting Corpus Christi next Tuesday.

Table 1: Example sentences used for data collection.

board and keystroke inputs respectively. Therefore, there are altogether four input configurations. For each configuration, 250 sentences were collected from a non-native fluent English speaker. 200 sentences were used as test data while the remaining 50 sentences were used for acoustic model adaptation. These sentences contain a variety of given names, surnames and city names so that confusions cannot be easily resolved using a language model. Example sentences used for data collection are shown in Table 1. In order to investigate the robustness of HVR in noisy environment, the collected speech data were also artificially corrupted with additive babble noise from the NOISEX database (Varga and Steeneken, 1993) to synthesise noisy speech signal-to-noise (SNR) levels of 20 and 10 decibels[2].

The ASR system used in all the experiments reported in this paper consists of a set of HMM-based triphone acoustic models and an n-gram language model. The HMM models were trained using 39-dimensional MFCC features. Each HMM has a left-to-right topology and three emitting states. The emission probability for each state is represented by a single Gaussian component [3]. A bigram language model with a vocabulary size of 200 words was used for testing. The acoustic models were also noise-compensated using VTS (Acero et al., 2000) in order achieve a better baseline performance.

6.1 Comparison of Input Speed

Table 2 shows the speech, letter and total input speed using different input configurations. For synchronous HVR, the total input speed is the same as the speech and letter input speed since both the speech and haptic inputs are provided concurrently. According to this study, synchronous keyboard input speed is 86 words per minutes (WPM). This is slightly faster than keystroke input using handwriting gestures, where the input speed is 78 WPM. This is not surprising since key taps are much quicker to generate compared to handwriting gestures. On the other hand, the individual speech and letter input speed are faster for asynchronous mode because users do not need to multi-task. However, since the speech and haptic inputs are provided concurrently, the resulting total input speed for asynchronous HVR is much slower compared to synchronous HVR. Therefore, synchronous HVR is potentially more efficient than asynchronous HVR.

Haptic Input	HVR Mode	Input Speed (WPM)		
		Speech	Letter	Total
Keyboard	Sync	86	86	86
	ASync	100	105	51
Keystroke	Sync	78	78	78
	ASync	97	83	45

Table 2: Comparison of the speech and letter input speed, measured in Words-Per-Minute (WPM), for different HVR input configurations

6.2 Performance of ASR

HVR Mode	SNR	WER (%)	LER (%)
ASync	Clean	22.2	17.0
	20 dB	30.2	24.2
	10 dB	33.3	28.5
Sync (Keyboard)	Clean	25.9	20.2
	20 dB	34.6	28.8
	10 dB	35.5	29.9
Sync (Keystroke)	Clean	29.0	22.5
	20 dB	40.1	32.0
	10 dB	37.9	31.3

Table 3: WER and LER performance of ASR in different noise conditions

First of all, the Word Error Rate (WER) and Letter Error Rate (LER) performances for standard ASR systems in different noise conditions are summarized in Table 3. These are results using pure ASR, without adding the haptic inputs. Speech recorded using asynchronous HVR is considered normal speech. The ASR system achieved 22.2%, 30.2% and 33.3% WER in clean, 20dB and 10dB

[2]Higher SNR indicates a better speech quality

[3]A single Gaussian component system was used as a compromise between speed and accuracy for mobile apps.

conditions respectively. Note that the acoustic models have been compensated using VTS (Acero et al., 2000) for noisy conditions. Table 3 also shows the system performance considering on the initial letter sequence of the recognition output. This indicates the potential improvements that can be obtained with the additional first letter information. Note that the pure ASR system output contains substantial initial letter errors.

For synchronous HVR, the recorded speech is expected to exhibit different characteristics since it may be influenced by concurrent haptic input. Table 3 shows that there are performance degradations, both in terms of WER and LER, when performing ASR on these speech utterances. Also, the degradations caused by simultaneous keystroke input are greater. The degradation may be caused by phenomena such as the presence of *filled pauses* and the *lengthening* of phoneme duration. Other forms of disfluencies may have also been introduced to the realized speech utterances. Nevertheless, the additional information provided by the PLIs will outweigh these degradations.

6.3 Performance of Synchronous HVR

Haptic Input	SNR	WER (%)	LER (%)
Keyboard	Clean	11.8	1.1
	20 dB	12.7	1.0
	10 dB	15.0	1.0
Keystroke	Clean	11.4	0.3
	20 dB	13.1	0.9
	10 dB	14.0	1.0

Table 4: WER and LER performance of synchronous HVR in different noise conditions

The performance of synchronous HVR is shown in Table 4. Compared to the results shown in Table 3, the WER performance of synchronous HVR improved by approximately a factor of two. Furthermore, the LER performance improved significantly. For keyboard input, the LER reduced to about 1.0% for all noise conditions. Note that the tradeoffs between the WER and LER performance can be adjusted by applying appropriately weights to different knowledge sources during integration. For keystroke input, top five letter candidates returned by the handwriting recognizer were used. Therefore, in clean condition, the acoustic models are able to recover some of the errors introduced by the handwriting recognizer, bringing the LER down to as low as 0.3%. However, in noisy conditions, the LER performance is similar to those using keyboard input. Overall, synchronous and asynchronous HVR achieved WER comparable performance.

6.4 Performance of Asynchronous HVR

Haptic Input	SNR	WER (%)	LER (%)
Keyboard	Clean	10.2	0.6
	20 dB	11.2	0.6
	10 dB	13.0	0.6
Keystroke	Clean	10.7	0.4
	20 dB	11.4	1.0
	10 dB	13.4	1.1

Table 5: WER and LER performance of asynchronous HVR in different noise conditions

Similar to synchronous HVR, asynchronous HVR also achieved significant performance improvements over the pure ASR systems. Table 5 shows the WER and LER performance of asynchronous HVR in different noise conditions. The WER performance of asynchronous HVR is consistently better than that of synchronous HVR (comparing Tables 4 and 5). This is expected since the speech quality for asynchronous HVR is higher. However, considering the much slower input speed (*c.f.* Table 2) and the marginal WER improvements for asynchronous HVR, synchronous HVR appears to be a better configuration.

6.5 Integrated Decoding vs. Lattice Rescoring

SNR	WER (%)		
	Clean	20dB	10dB
Integrated	11.8	12.7	15.0
Lat-rescore	11.2	18.6	18.1

Table 6: WER performance of keyboard synchronous HVR using integrated decoding and lattice rescoring

As previously mentioned in Section 5, HVR can also be performed in two stages using lattice rescoring technique. Table 6 shows the performance

comparison between integrated decoding and lattice rescoring for HVR. Both methods gave similar performance in clean condition. However, lattice rescoring yielded significantly worse performance in noisy environment. Therefore, it is important to tightly integrate the PLI into the decoding process to avoid premature pruning away optimal paths.

6.6 Runtime Performance

ASR system searches for the best word sequence using a dynamic programming paradigm (Ney and Ortmanns, 1999). The complexity of the search increases with the vocabulary size as well as the length of the input speech. A well-known concept of *Token Passing* (Young et al., 1989) can be used to describe the recognition search process. A set of active tokens are being propagated upon observing an acoustic feature frame. The best token that survived to the end of the utterance represents the best output. Typically, beam pruning technique (Ortmanns et al., 1997) is applied to improve the recognition efficiency. Tokens which are unlikely to yield the optimal solution will be pruned away. HVR performs a more stringent pruning, where paths that do not conform to the PLI sequence are also be pruned away.

System	SNR	RT	Active Tokens Per Frame
ASR	Clean	1.9	6260
	20 dB	2.0	6450
	10 dB	2.4	7168
Keyboard	Clean	0.9	3490
	20 dB	0.9	3764
	10 dB	1.0	4442
Keystroke	Clean	1.1	4059
	20 dB	1.2	4190
	10 dB	1.5	4969

Table 7: WER and LER performance of integrated and rescoring synchronous HVR in different noise conditions

Table 7 shows the comparison of the runtime factors and the average number of active tokens per frame for ASR and HVR systems. The standard ASR system runs at 1.9, 2.0 and 2.4 times real-time (xRT)[4]. The runtime factor increases with decreasing SNR because the presence of noise introduces more confusions, which renders beam pruning (Ortmanns et al., 1997) less effective. The number of active tokens per frame also increases from 6260 to 7168 as the SNR drops from the clean condition to 10dB. On the other hand, there are significant speedup in the runtime of HVR systems. In particular, synchronous HVR achieved the best runtime performance, which is roughly consistent across different noise conditions (approximately 1.0 xRT). The average number of active tokens also reduces to the range of 3490 – 4442. Therefore, the synchronous HVR using keyboard input is robust to noisy environment, both in terms of WER and runtime performance. The runtime performance using keystroke input is also comparable to that using keyboard input (only slightly worse). Therefore, both keyboard and keystroke inputs are effective ways for entering the initial letters for HVR. However, it is worth noting that the iPad was used for the studies conducted in this work. The size of the iPad screen is sufficiently large to allow efficient keyboard entry. However, for devices with smaller screen, keystroke inputs may be easier to use and less error-prone.

[4]Runtime factor is computed as the ratio between the recognition duration and the input speech duration

7 Conclusions

This paper has presented a unifying probabilistic framework for the multimodal Haptic Voice Recognition (HVR) interface. HVR offers users the option to interact with the system using touchscreen during voice input so that additional *cues* can be provided to improve the efficiency and robustness of voice recognition. Partial Lexical Information (PLI), such as the initial letter of the words, are used as cues to guide the recognition search process. Therefore, apart from the acoustic and language models used in conventional ASR, HVR also combines the haptic model as well as the PLI model to yield an integrated probabilistic model. This probabilistic framework integrates multiple knowledge sources using the weighted finite state transducer operation. Such integration is achieved using the composition operation which can be applied on-the-fly to yield efficient implementation. Experimental results show that this framework can be used to achieve a more efficient and robust multimodal interface for text entry on modern portable devices.

References

Alex Acero, Li Deng, Trausti Kristjansson, and Jerry Zhang. 2000. HMM adaptation using vector Taylor series for noisy speech recognition. In *Proc. of ICSLP*, volume 3, pages 869–872.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner. 2005. An empirical study of typing rates on mini-qwerty keyboards. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1288–1291, New York, NY, USA. ACM.

S. B. Davis and P. Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 28(4):357–366.

Mark Gales and Steve Young. 1996. Robust continuous speech recognition using parallel model combination. *IEEE Transactions on Speech and Audio Processing*, 4:352–359.

H. Hermansky. 1990. Perceptual Linear Predictive (PLP) analysis of speech. *Journal of the Acoustic Society of America*, 87(4):1738–1752.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

H. Ney and S. Ortmanns. 1999. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83.

J. Ortega-Garcia and J. Gonzalez-Rodriguez. 1996. Overview of speech enhancement techniques for automatic speaker recognition. In *Proceedings of International Conference on Spoken Language (ICSLP)*, pages 929–932.

S. Ortmanns, H. Ney, H. Coenen, and Eiden A. 1997. Look-ahead techniques for fast beam search. In *ICASSP*.

L. A. Rabiner. 1989. A tutorial on hidden Markov models and selective applications in speech recognition. In *Proc. of the IEEE*, volume 77, pages 257–286, February.

K. C. Sim. 2010. Haptic voice recognition: Augmenting speech modality with touch events for efficient speech recognition. In *Proc. SLT Workshop*.

A. Varga and H.J.M. Steeneken. 1993. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Communication*, 12(3):247–251.

S.J. Young, N.H. Russell, and J.H.S Thornton. 1989. Token passing: a simple conceptual model for connected speech recognition systems. Technical report.