Journal of VLSI Signal Processing

DOI: 10.1007/s11265-005-4150-4

A Speech-Centric Perspective for Human-Computer Interface: A Case Study

LI DENG AND DONG YU
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Received xx, xx; Accepted xx, xx

Published online: xx xx

Abstract. Speech technology has been playing a central role in enhancing human-machine interactions, especially for small devices for which graphical user interface has obvious limitations. The speech-centric perspective for human-computer interface advanced in this paper derives from the view that speech is the only natural and expressive modality to enable people to access information from and to interact with any device. In this paper, we describe some recent work conducted at Microsoft Research, aimed at the development of enabling technologies for speech-centric multimodal human-computer interaction. In particular, we present a case study of a prototype system, called MapPointS, which is a speech-centric multimodal map-query application for North America. This prototype navigation system provides rich functionalities that allow users to obtain map-related information through speech, text, and pointing devices. Users can verbally query for state maps, city maps, directions, places, nearby businesses and other useful information within North America. They can also verbally control applications such as changing the map size and panning the map moving interactively through speech. In the current system, the results of the queries are presented back to users through graphical user interface. An overview and major components of the MapPointS system will be presented in detail first. This will be followed by software design engineering principles and considerations adopted in developing the MapPointS system, and by a description of some key robust speech processing technologies underlying general speech-centric human-computer interaction systems.

Keywords: human-computer interaction, speech-centric multimodal interface, robust speech processing, MapPointS, speech-driven mobile navigation system

1. Introduction

Speech recognition technology enables a computer to automatically convert an acoustic signal uttered by users into textual words, freeing them from the constraints of the standard desktop-style interface (such as mouse pointer, menu, icon, and window etc.). The technology has been playing a key role in enabling and enhancing human-machine communications. In combination with multimedia and multimodal processing technologies, speech processing will in the future also contribute, in a significant way, to facilitating human-human interactions. In applications such as distributed meetings, audio-visual browsing, and multimedia annotations, automatic processing of natural, spontaneous speech will collaborate with automatic audio-visual object tracking and other multimedia processing techniques to complete full end-to-end systems. In addition to the multimedia applications, the most important role that speech can play is in a full range of the devices that demand efficient human inputs. Since speech is the only natural and expressive modality for information access from and interaction with any device, we highlight the speech-centric view of human-machine interface (HCI).

Speaking is the most natural form of human-to-human communication. We learn how to speak in the childhood, and we all exercise our speaking communication skills on a daily basis. The possibility to translate this naturalness of communication into the capability of a computer is our natural expectation, since a computer is indeed equipped with huge computing and storage capacities. However, the expectation that computers should be good at speech has not been a reality, at least not yet. One important reason for this is that speech input is prone to error due to imperfection of the technology in dealing with variabilities from the speaker, speaking style, and the acoustic environment. The imperfection, in addition to a number of social and other reasons, raises the issue that speech alone is not sufficient as the most desirable input to computers. Use of multimodal inputs in an HCI system, which fuses two or more input modalities (speech, pen, mouse, etc.) to overcome imperfection of speech technology in its robustness as well as to complement speech input in other ways, is becoming an increasingly more important research direction in HCI.

Major HCI modalities in addition to speech are related to graphic user interface (GUI). GUI is based primarily on the exploitation of visual information, and has significantly improved HCI by using intuitive real-world metaphors. However, it is far from the ultimate goal of allowing users to interact with computers without training. In particular, GUI relies heavily on a sizeable screen, keyboard, and pointing device, which are not always available. In addition, with more and more computers designed for mobile usages and hence subject to the physical size and hands-busy or eyes-busy constraints, the traditional GUI faces an even greater challenge. Multimodal interface enabled by speech is widely believed to be capable of dramatically enhancing the usability of computers because GUI and speech have complementary strengths. While speech has the potential to provide a natural interaction model, the ambiguity of speech and the memory burden of using speech as output modality on the user have so far prevented it from becoming the choice of mainstream interface. Multimodal Intelligent Personal Assistant Device, or MiPad, was one of our earlier attempts in overcoming such difficulties by developing enabling technologies for speech-centric multimodal interface. MiPad is a prototype of wireless Personal Digital Assistant (PDA) that enables users to accomplish many common tasks using a multimodal spoken language interface (speech + pen + display). MiPad, as a case study for speech-centric multimodal HCI application, has been described in detail in our recent publication [2]. In this paper, we will present a second case study based on a new system built within our research group more recently, called MapPointS.

During past several years, many different methods of integrating multiple modalities (voice, visual, and others) in HCI have been proposed and implemented, and some key issues have been discussed [10–13, 16]. Many prototype systems have also been built based on the use of multiple modalities [1, 2, 7, 9, 14], most of which have focused on the special advantage of the speech input for mobile or wireless computing as in multimodal PDA's. Both of our prototype systems, MiPad and MapPointS, have such mobile computing in the special design consideration. Their design also takes the speech-centric perspective — fully exploiting the efficiency of the speech input where other modalities have special difficulties.

The focus of this paper, the prototype MapPointS, is a speech-centric, multimodal, location-related, map-query application for North America. The unique advantage of the system is its full and direct exploitation of the frequently updated backend database provided by the existing Microsoft product, MapPoint (http://mappoint.msn.com). MapPointS essentially adds the "Speech" modality and its interface into MapPoint, and hence MapPointS. MapPointS provides rich functionalities to allow the users to obtain map-related information through speech, text, and pointing devices. (MapPoint provides the same functionalities with the inputs of text and pointing devices only). With MapPointS, the users can verbally query for state maps, city maps, directions, places (e.g., school names), nearby businesses, and many other useful information. They can also verbally control applications such as changing the map size and panning the map moving interactively through speech. In the current system, the results of the queries are presented back to users through GUI. An overview and the major components of the MapPointS system will be presented in detail in this paper first. Following this presentation, we will describe several key software design engineering principles and considerations in developing MapPointS. Finally we will present some key speech processing technologies underlying the general speech-centric HCI systems including MiPad and MapPointS.

Journal of VLSI Signal Processing SJNW437-04-4150 October 1, 2005 3:23

2. System Overview and Functionality of Mappoints

MapPointS is a map query application that supports a large set of map query commands through speech, text, and pointing devices. These commands can be classified into the following five categories:

1. *Application Control*: Application control commands are used to control MapPointS applications. For example, a user can use speech (as well as other modalities) to quit the application, to pan the map towards eight directions, to zoom the maps, or to open and save the map.
2. *Location Query*: Location queries are used to search for the map of a specific location. For example, a user can query for a map with city names, state names, joint city and state names, place names (e.g., Seattle University), or referenced locations (e.g., here; this place; and this area, etc., which are indicated by the mouse click rather than by the speech input.
3. *Route Query*: Route queries are used to obtain directions from one location to another. There are two types of such queries. The first type contains both "from" and "to" information. For example, a user can say "How do I get from <startlocation> to <endlocation>" to obtain directions from <startlocation> to <endlocation>. The <startlocation> and <endlocation> can be any location type specified in location query. The second type of queries contains information about "to location" only. "How may I go to <location>" is an example of such queries. When a query with "to location" only is submitted by a user, the system will infer the most probable from location based on the user's dialog context.
4. *Nearest Query*: "Nearest" queries are used to find the closest or the nearest instance of a specific type of places to the current location. MapPointS supports about 50 types of locations including bank, gas station, airport, ATM machine, restaurant, and school. For instance, a user can query for the nearest school, Chinese restaurant, etc. When such a query is made, MapPointS will infer the most probable current reference location based on the dialog context.
5. *Nearby Query*: "Nearby" queries are similar to the "nearest" queries above. The difference is that all nearby instances of a type of places, instead of only one, are displayed in the nearby queries. For example, a user can query for all nearby gas stations. Similar to the situation of the nearest query, MapPointS needs to infer the most probable reference location before executing the query.

Examples of the above five types of queries are provided now. Figure 1 is a screen shot where a map of Seattle is displayed as a result of speech command used in the location query: "show me a map of Seattle". A typical map of Seattle with its surroundings is immediately displayed. All cities in the U.S. can be queries in the same manner.

Figure 2 gives a multimodal interaction example where the user makes a location query by selecting an area with mouse and zooming the picture to just that part of the map while using the following simultaneous speech command: "show me this area". The portion of the map selected by the user is displayed in response to such a multimodal query.

In Fig. 3 is another multimodal interaction example for the nearest location query. In this case, the user clicks on a location, and more or less simultaneously issues the command: "Show me the nearest school" with speech. MapPointS displays "Seattle University" as the result based on the location that the user just clicked on.

In Fig. 4 we show an example of the route query to find the direction from Seattle to Boston, with a speech utterance such as "Show me directions from Seattle to Boston", or "How may I go from Seattle to Boston", etc. If the immediately previous location is Seattle, then saying just "How may I go to Boson" will give the identical display as the response to the query.

We provide a further example in Fig. 5 of querying nearby restaurants by speaking to MapPointS with "show me all nearby restaurants". The system assumes the current location of the user based on the previous interactions, and is hence able to display all nearby restaurants without the need for the user to specify where he currently is.

For the system functionalities illustrated in the above description and examples, MapPointS demonstrates the following four specific features:

1. *Multi-Modal Human-Computer Interaction*: As we discussed in Introduction section, one of the trends of HCI is the integration of multi-modal inputs, through which speech recognition is integrated with various other modalities such as keyboard and

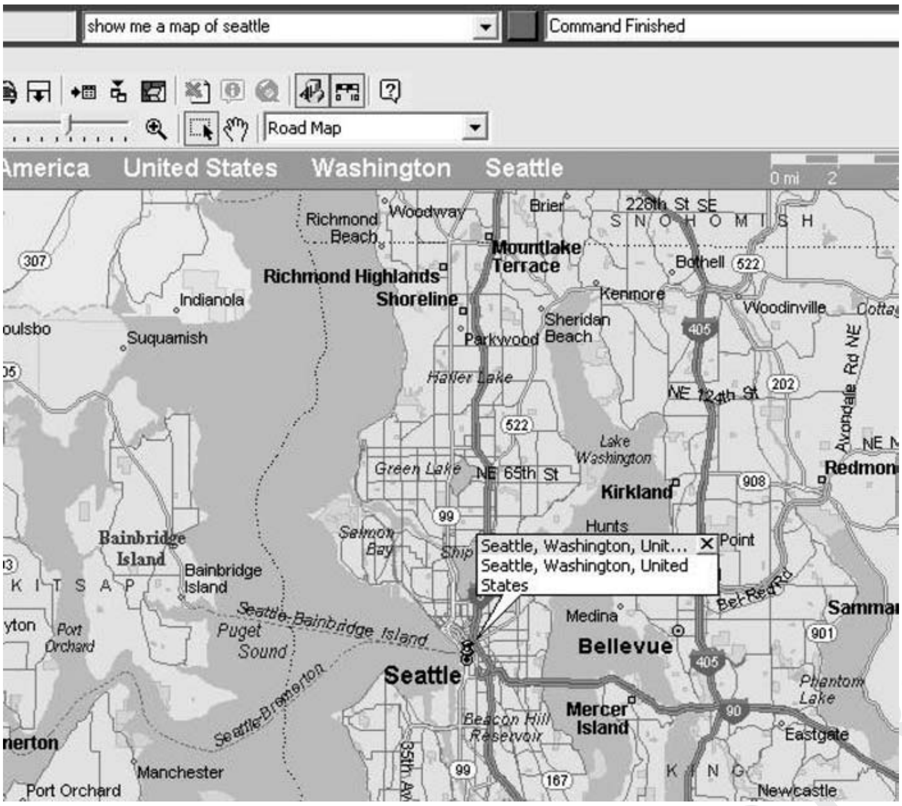

Figure 1. Navigation using voice command: “show me a map of Seattle”.

mouse inputs. MapPointS is a good show case for this capability since it includes both location search (via the name) and location pointing/selection. The former is most naturally accomplished using voice command because it is difficult to use a mouse or a pen to search for one of a very large number of items (cities, etc). The latter, location pointing and selection, on the other hand, is relatively easy to be fulfilled with mouse clicks. For example, a user may ask the system to “show me a map of Seattle”. The user can then use the mouse to click on a specific location or to select a specific area. He/she can then or simultaneously issue the command “Show me the nearest school around here” with speech as the input.

2. *Integrated Interface for Speech and Text*: In the MapPointS, a user not only can use speech to query the application but also can use a natural text input to ask for the same thing. For example, the user can say “Where is the University of Washington” to have the University of Washington be identified in the map. Alternatively, the user can just type in “Where is the University of Washington” in the command bar and obtain the same result.
3. *Recognition of a Large Quantity of Names*: As we have mentioned, MapPointS allows its users to query for all cities and places in the US. Accurate recognition of all these names is difficult since there are too many names to be potential candidates. For example, there are more than 30,000 distinct city names in the US, and the total number of valid combinations of “city, state” alone is already larger than 100,000, not to mention all the school names, airport names, etc. in all cities.
4. *Inference of Missing Information*: When a user queries information, he/she may not specify full information. For example, when a user submits a query “How may I get to Seattle University”, MapPointS needs to infer the most probable location that the user is currently at. This inference is automatically performed based on the previous interactions between the user and MapPointS.

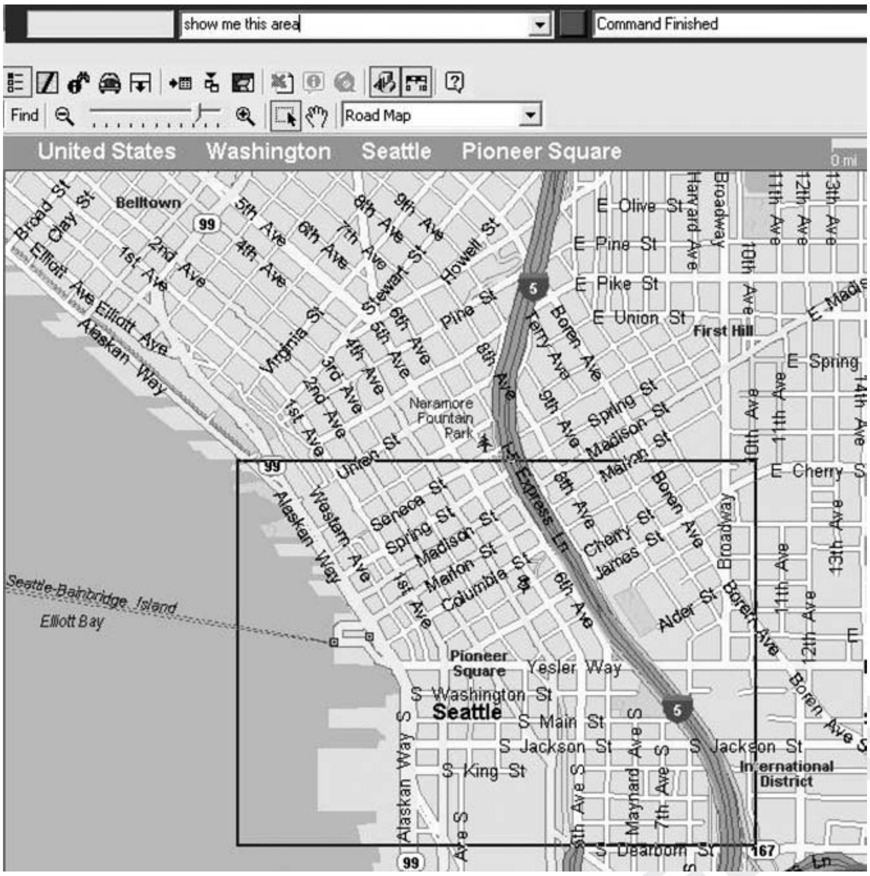

Figure 2. User's mouse selection is seamlessly integrated into the speech command: "Show me this area".

3. System Architecture and Components of Mappoints

The major system components of MapPointS are depicted in Fig. 6. The raw signals generated by the user are first processed by a semantic parser into the "*surface semantics*" representation. For the speech input, the speech recognizer first converts the raw signal into a text sequence, with the help from the Language Model component, before semantic parsing. Each possible modality, speech or otherwise, has its separate corresponding semantic parser. However, the resulting surface semantics are represented in common Semantic Markup Language (SML) format and is thus independent of the modality. With this approach, the input methods become separated from the rest of the system. The surface semantics from all the input media are then merged by the Discourse Manager component into the "discourse semantics" representation. When generating the discourse semantics, the discourse manager integrates the environment information (provided by the Environment Manager and Semantic Model components) which includes: (1) dialog context; (2) domain knowledge; (3) user's information, and (4) user's usage history. Such important environment information is used to adapt the Language Model, which improves the speech recognition accuracy and enhance the Semantic Parsers for either the speech or text input. (Semantic Model is the component that provides rules to convert the surface semantics into actionable commands and to resolve possible confusibility.) The discourse semantics is then fed into the Response Manager component to communicate back to the user. The Response Manager synthesizes the proper responses, based on the discourse semantics and the capabilities of the user interface, and plays the response back to the user. In this process, Behavior model provides rules to carry out the required actions.

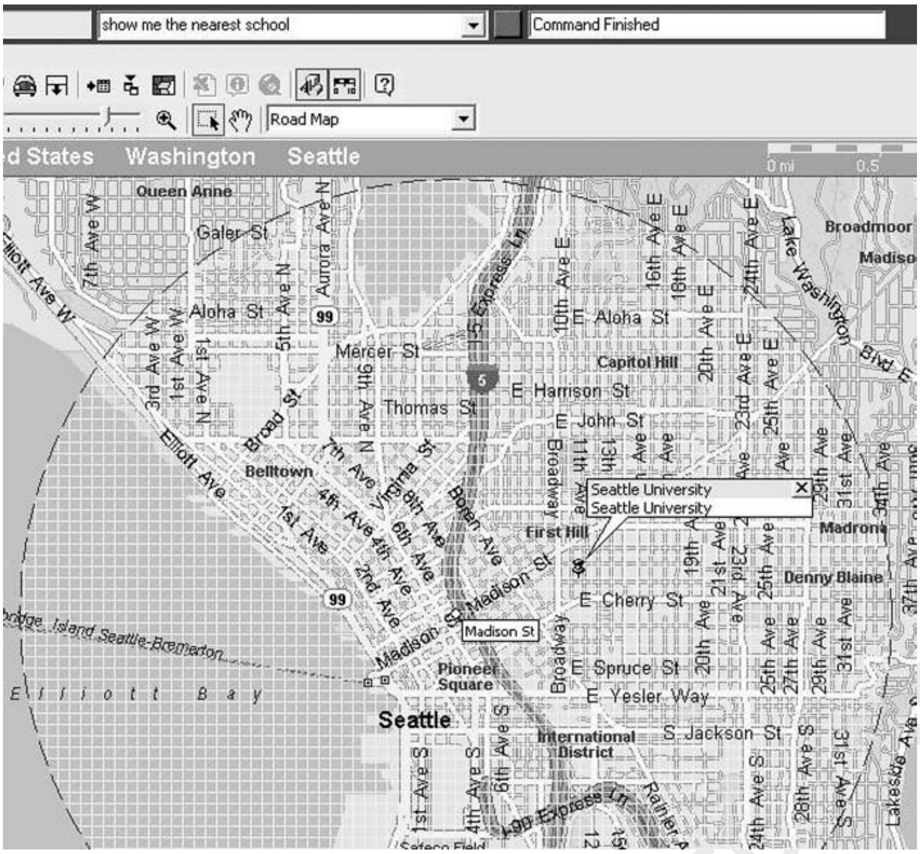

Figure 3. User's latest mouse click input is referenced by voice command: "Show me the nearest school".

We have already introduced some components of the above main architecture in some of our earlier publications (e.g., [2]). In this paper, we focus on two novel components of the architecture: Language Model (LM) and Environment Manager. The design of these two components has been specific to the MapPointS system.

As we pointed out in the previous section, one of the major difficulties of the task is the recognition of the very large quantity of names. Including all names in the grammar is infeasible because the total number of names is so large that the confusability between these names is extremely high and the computation for speech recognition search is very expensive.

The speech recognition task is conducted as an optimization problem to maximize the posterior probability:

$$\hat{w} = \arg\max_{w} P(A \mid w)P(w),$$

where w is a candidate word sequence, and $P(w)$ is the prior probability for the word sequence (or LM probability). This suggests that we can reduce the search effort through controlling the language model so that only the most probable names are kept in the search space. One of the approaches used to better estimate $P(w)$ is to exploit the user information, especially the user's home address, usage history, and current location. In other words, we can simplify the speech recognition search task by optimizing the following posterior probability:

$$\hat{w} = \arg\max_{w} P(A \mid w)P(w \mid E),$$

where the general LM $P(w)$ is now refined (i.e., adapted) to the Environment-specific LM $P(w \mid E)$, which has a much lower perplexity than the otherwise generic LM. (This environment-specific LM is closely related to topic-dependent LM or user-adapted LM in the literature.) How to exploit the user

Figure 4. Route query to find direction from Seattle to Boston by speaking to MapPointS: "How may I go from Seattle to Boston", or just "How may I go to Boston" if the current location is Seattle.

"environment" information to adapt the LM is the job of the "Environment Manager" component in Fig. 1, which we describe in detail in the remainder of this section.

In the current MapPointS system, the PCFG (Probabilistic Context Free Grammar) is used as the LM. Some examples of the CFG rules are shown below:

<query> ➔ <app_query> | <pan_query> | <zoom_query> | <location_query> |<route_query>| <nearest_query> | <nearby_query> | …

<location_query> ➔ show me <location> | show me a map of <location> |where is <location> | …

<location> ➔ <pointed_location> | <named_location> | …

<pointed_location> ➔ here | this point | this | this place | …

<named_location> ➔ <city> | <state> |<city_state> | <well-known_place> | …

<city> ➔ New York City | Seattle | Dallas | …

In order to build the environment-adapted LM based on the PCFG grammar, the LM probability $P(w \mid E)$ is decomposed into the product of the words that make up the word sequence w and that follow the grammar at the same time. The majority of the words which are relevant to LM in our MapPointS system are the names or name phrases such as "**New York City**" in the above CRG rules. (Many non-name words in the grammar are provided with uniform LM probabilities and hence they become irrelevant in speech recognition and semantic parsing.)

We now describe how the conditional probability of a name or name phrase given the environment (user) information is computed by the Environment Manager component of MapPointS. Several related conditional probabilities are computed in advance based on well motivated heuristics pertaining to the MapPointS task. First, it is noted that users tend to move to a city before querying for small and less-known locations inside that city. On the other hand, they often move between cities and well-known places at any time. In other words, small places (such as restaurants) in a city, except for the city that the user is looking at currently, have very small prior probabilities. Cities, well-known places, and small places in the currently visited city, in contrast, have much higher prior probabilities. For this reason, we organize all names into two categories: the global level and the local level. The global-level name list contains state names, city names, City+State, and well-known places such as Yellowstone National

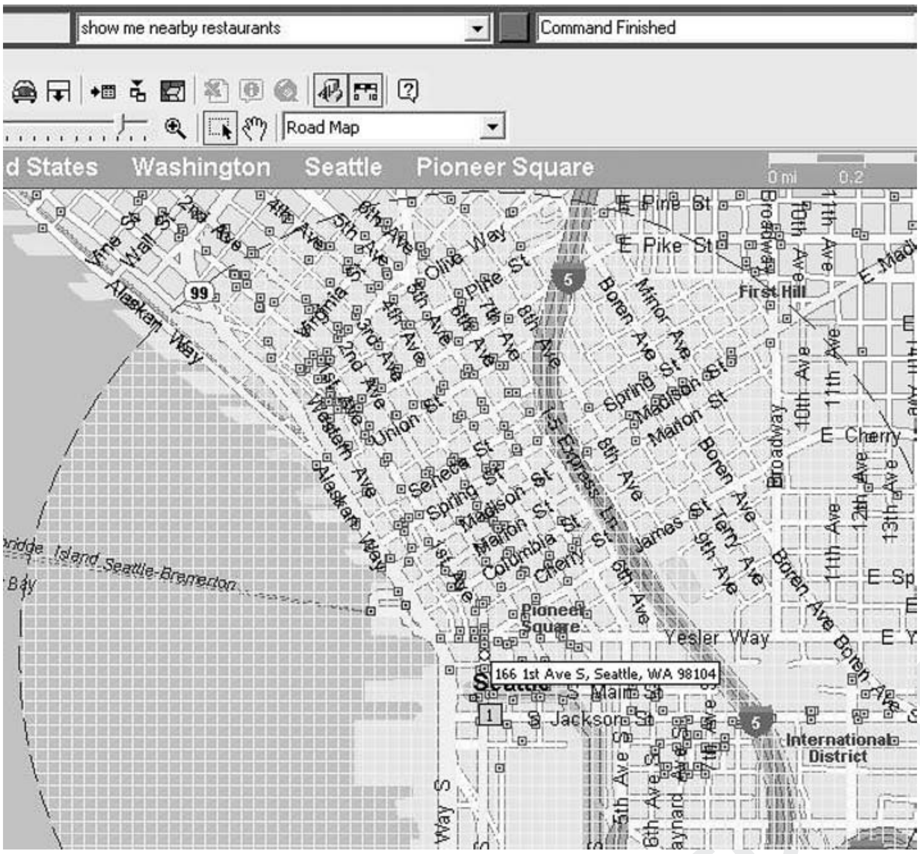

Figure 5. Display of MapPointS in response to the "Nearby Restaurants" query.

park. This global-level name list is included in the recognition grammar at all times. The local-level name list, on the other hand, contains detailed location information about a city or a well-known place. When the current city is changed, the local-level name list is changed accordingly.

To speed up the loading of the local-level name list, we pre-built the local list for each of the 2000 major cities. This is needed because there are usually many place names in large cities and these lists are slow to build. For local-name lists of small cities, we build them when the city is firstly visited and cache the lists in the hard drive in order to speed up the process when it is visited again.

Even after adopting this approach, the number of names is still large. The majority of the names in the global-level name list are for cite and state combination (City+State). The simplest way to include these names in the grammar would be to list them all one by one. This, however, requires more than 100,000 distinct entries in the grammar. Typical recognition engines can not handle the grammars of such a size efficiently and effectively. We thus take a further approach to arrange the cities and states in separate lists and allow for combinations of them. This approach greatly reduces the grammar size since we only need 30,000 cities and 50 states. Unfortunately, this will provide invalid combinations such as "Seattle, California". It also increases the name confusability since now there are more than $30{,}000 * 50 = 1{,}500{,}000$ possible combinations. To overcome this difficulty, we choose to list only valid City+State combinations. To accomplish this, we prefix the grammar so that all names are organized based on the city names, and each city name can only follow the valid subset of the 50 state names. The prefixed grammar can be processed by recognition engines rather efficiently. For some slow systems where the speed and accuracy may still be inadequate, we further pruned the number of City+State combinations.

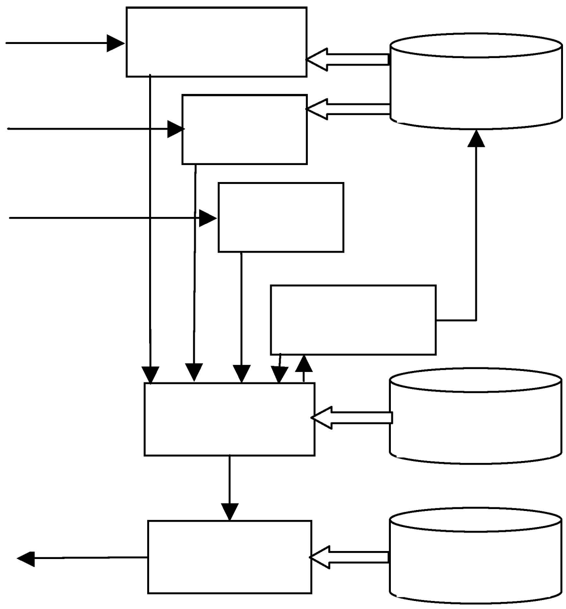

Figure 6. Major system architecture and components in MapPointS.

The second heuristic adopted by the MapPointS system is motivated by the intuition that if a user queries restaurants a lot, the probability that he/she will query new restaurants should be high even though they have not been queried before. With this heuristic, we organize all names into about 40 classes including gas stations, schools, restaurants, airports, etc. A complete list of the classes can be found in Table 1.

We denote the probability that a class of names is queried as $P([\text{Class}]|\text{History})$ or $P([C]|H)$. The estimate for this probability is provided as in the MapPointS system:

$$P([C_i] \mid H) = \frac{\sum_k \exp(-\lambda_h(T - t_{ik}))}{\sum_j \sum_k \exp(-\lambda_h(T - t_{jk}))}$$

where t_{ik} is the time the names in class C_i was queried the k-th time (as the "History" information), T is the current time, and λ_h is the forgetting factor. We further assume that $[C_i]$ is independent of other factors in the environment. This particular form of the probability we have adopted says that the further away a past class query is, the less it will contribute to the probability of the current class query.

The third heuristic we have adopted is motivated by the intuition that even though names in the global-level name list are likely to be queried by users, the probabilities that each name would be queried will be different. For example, large cities such as San Francisco and Boston are more likely to be queried than small cities such as Renton. For this reason, we estimated the prior probabilities of all cities and

Table 1. Full list of location classes in MapPointS.

Class ID	Class Type
1	State
2	City
3	Well-known Places
4	Galleries
5	ATMs and banks
6	Gas stations
7	Hospitals
8	Hotels and motels
9	Landmarks
10	Libraries
11	Marinas
12	Museums
13	Nightclubs and taverns
14	Park and rides
15	Police stations
16	Post offices
17	Rental car agencies
18	Rest areas
19	Restaurants—Asian
20	Restaurants—Chinese
21	Restaurants—delis
22	Restaurants—French
23	Restaurants—Greek
24	Restaurants—Indian
25	Restaurants—Italian
26	Restaurants—Japanese
27	Restaurants—Mexican
28	Restaurants—pizza
29	Restaurants—pizza
30	Restaurants—seafood
31	Restaurants—Thai
32	Schools
33	Shopping
34	Casinos
35	Stadiums and arenas
36	Subway stations
37	Theaters
38	Airports
39	Zoos

well-known places in advance. The estimation is based on the MapPoint.NET (http://mappoint.msn.com/) IIS (Internet Information Server) log data. The IIS log records raw queries users of the MapPoint.NET submitted (The log, however, does not contain any user identification information).

We processed more than 40GB of the log data to obtain statistics of states, cities, and well-known places that users have queried. We found that for the cities, the probability computed by the log data is quite similar to that estimated based on the city population. We denote the probability for each name in the class given the class label as $P(N|[C]$; examples are P(Name|[Class]='City') and P(Name|[Class]='Well-KnownPlace'). For local-level names, we assume a uniform distribution for $P(\mathrm{N}|[C])$. Tables 2 and 3 show the most frequently queried 10 States and cities respectively:

The fourth heuristic implemented in the MapPointS system uses the intuition that location names related to the user are more likely to be queried than other names. For example, if a user lives in the Seattle, he/she is more likely to query locations in or close to the Seattle. We calculate this probability class by class. We denote this probability as P(Name|[Class],User) or simply $P(N|[C],U)$ and estimate it according to:

$$P(N_i \mid [C_k], U) = \frac{S(N_i \mid [C_k], U)}{\sum_{j:N_j \in [C_k]} S(N_j \mid [C_k], U)}$$

where

$$S(N_i \mid [C_k], U) = \exp\left(-\lambda_u d_{iU}\right) P(N_i \mid [C_k]),$$

Table 2. Top 10 States queried by users of MapPoint.NET and their estimated probabilities.

Top no.	Name	Occurrence in IIS log	Relative frequency
1	California	2950295	0.127832
2	Texas	1791478	0.009605
3	Florida	1512045	0.065515
4	New York City	1117964	0.048440
5	Pennsylvania	1074052	0.046537
6	Illinois	1024543	0.044392
7	Ohio	1006874	0.043626
8	New Jersey	782871	0.033920
9	Michigan	776841	0.033660
10	Georgia	738660	0.032005

Table 3. Top 10 cities queried by users of MapPoint.NET and their estimated probabilities.

Top #	Name	Occurrence in IIS log	Relative Frequency
1	Houston, Texas	309246	0.014637
2	Chicago, Illinois	202948	0.009605
3	Dallas, Texas	169710	0.008032
4	Los Angeles, California	166005	0.007857
5	San Diego, California	141622	0.006656
6	Atlanta, Georgia	140637	0.006656
7	Orlando, Florida	135911	0.006433
8	San Antonio, Texas	122723	0.005809
9	Seattle, Washington	115550	0.005469
10	Las Vegas, Nevada	113927	0.005392

and d_{iU} is the distance between $N_i \in C_k$ and the user's home. λ_u is the corresponding decaying parameter.

The fifth heuristic uses the intuition that locations close to the currently visited city are more likely to be queried than other locations. Following the same example, if the user lives in Seattle, not only is he/she more likely to query Bellevue than Springfield, but he/she is also more likely to query for "Everett, Washington" than "Everett, Massachusetts". We denote this probability as P(Name|[C],CurrentLocation) or simply $P(N|[C], L)$ and estimate it as:

$$P(N_i \mid [C_k], L) = \frac{S(N_i \mid [C_k], L)}{\sum_{j:N_j \in C_k} S(N_j \mid [C_k], L)}$$

where

$$S(N_i \mid [C_k], L) = \exp(-\lambda_l d_{iL}) P(N_i \mid [C_k]),$$

and d_{iL} is the distance between $N_i \in C_k$ and the current location. λ_l is the corresponding decaying factor.

The final, sixth heuristic we adopted is based on the intuition that if a user queries a location often recently, he/she is likely to query the same location again in the near future. For example, if the user lives in Seattle, but he/she queried for "Everett, Massachusetts" several times recently, we would expect that he will more likely to query for "Everett, Massachusetts" than "Everett, Washington" even though Everett, Washington" is more close to his home. We denote this probability as $P(\text{Name}|[C],\text{History})$ or simply $P(N|[C],H)$ and estimate it as:

$$P\left(N_i \mid [C_n], H\right) = \frac{S\left(N_i \mid [C_n], H\right)}{\sum_{j:N_j \in C_n} S\left(N_i \mid [C_n], H\right)}$$

where

$$S(N_i \mid [C_n], H) = \sum_k \exp(-\lambda_h (T - t_{ik})) P(N_i \mid [C_n])$$

and t_{ik} is the time when the name $N_i \in C_n$ was queried the k-th time. T is the current time, and λ_h is the forgetting factor.

With the above assumptions and heuristics based on well founded intuitions, we obtain the conditional probability $P(\text{Name} \mid \text{Environment})$ as:

$$\begin{aligned} P(N_i \mid E) &= \sum_{C_n} P(N_i \mid [C_n], E) P([C_n] \mid E) \\ &= \sum_{C_n} P(N_i \mid [C_n], U, L, H) P([C_n] \mid H) \\ &= \sum_{C_{ni}} \frac{P(N_i, U, L, H \mid [C_n])}{P(U, L, H \mid [C_n])} P([C_n] \mid H) \\ &= \sum_{C_{ni}} \frac{P(U, L, H \mid N_i, [C_n]) P(N_i \mid [C_n])}{P(U, L, H \mid [C_n])} \\ &\quad \times P([C_n] \mid H) \end{aligned}$$

We further assume that U, L, and H are independent of each other. This leads to the approximation of

$$\begin{aligned} P(N_i \mid E) &\approx \sum_{C_{ni}} \frac{P\left(U \mid N_i, [C_n]\right) P\left(L \mid N_i, [C_n]\right) P\left(H \mid N_i, [C_n]\right) P\left(N_i \mid [C_n]\right)}{P\left(U \mid [C_n]\right) P\left(L \mid [C_n]\right) P\left(H \mid [C_n]\right)} P\left([C_n] \mid H\right) \\ &= \sum_{C_{ni}} \frac{P\left(N_i \mid U, [C_n]\right) P\left(N_i \mid L, [C_n]\right) P\left(N_i \mid H, [C_n]\right)}{P^2\left(N_i \mid [C_n]\right)} P\left([C_n] \mid H\right) \end{aligned}$$

We can further simplify the above equation by assuming that each name belongs to one class. This is accomplished by using the location in the map—the semantic meaning of the name as the unique identifier of the name. For example, Everett can mean "Everett, Washington", "Everett, Massachusetts", "Everett Cinema", and somewhere else. In our MapPointS system's grammar, we allow for several different kinds of Everett's; each of them, however, is mapped to a different location in the semantic model with a different probability. This treatment removes the class summation in the above and we have the final expression of the environment-specific name probability of:

$$\begin{aligned} P(N_i \mid E) \\ = \frac{P(N_i \mid U, [C_n]) P(N_i \mid L, [C_n]) P(N_i \mid H, [C_n])}{P^2(N_i \mid [C_n])} \\ \times P([C_n] \mid H), \end{aligned}$$

where $N_i \in C_n$ and where all the probabilities at the right hand side of the equation have been made available using the several heuristics described above.

In the previous discussion, we normalize probabilities for each individual conditional probability in the above equations. However, the normalization can be done at the last step. We also noted that the system is not sensitive to small changes of the probabilities. With this in mind, in the MapPointS implementation, we only updated the probabilities when the probability change becomes large. For example, when the current location is 10 miles away to the previous location, or there are 20 new queries in the history. For the same reason, the decaying parameters and forgetting parameters are determined heuristically based on the observations from the IIS log.

Another important issue in the MapPointS system's LM computation is smoothing of the probabilities since the training data is sparse. In the current system implementation, the probabilities are simply backed up to the uniform distribution when no sufficient amounts of training data are available.

With all the above environment or user-specific LM implementation techniques provided by the Environment Manager component in the MapPointS system, most ambiguities encountered by the system can be resolved. For example, when a user asks: "Where is Everett", the system will infer the most probable Everett based on the different LM probabilities for the different Everett's. In most cases, the most probable Everett is either the closest Everett or the frequently visited Everett. In case the system's guess is incorrect, the user can submit a new query which contains more detailed information in the query. For example, he/she can say "Where is Everett, Washington".

Table 4. Four conditions under which the LM of the MapPointS system is constructed and the LM perplexity associated with each condition.

Conditions	LM perplexity
Uniform probability for all city/place names	5748528
Two-level structure for cities and places, but using uniform probabilities for city names	98810
Same as above but using prior probabilities of city names	5426
Same as above but including user-specific information	241

Further, in addition to providing useful environmental or user information to infer the probabilities of queries in LM, the Environment Manager component of MapPointS also permits the inference of missing elements in users' queries to obtain the complete discourse semantic information. This aspect has been discussed in [17] in detail and will not be described here.

We now present some quantitative results to show how the user modeling strategy discussed so far in this section has contributed to the drastic improvement of the LM. In Table 4, we list the perplexity numbers of the LM with and without the use of the user-specific information. These perplexity numbers are based on four ways of constructing the MapPointS system with and without using the probabilities and using user modeling. A lower perplexity of the LM indicates a higher quality of the LM, which leads to a lower ambiguity and higher accuracy for speech recognition. We observe from here that the system utilizing the user-specific information gives a much lower perplexity and better LM quality than that otherwise.

4. Software Engineering Considerations in Mappoints System Design

MapPointS involves its input from multiple modalities, its output in map presentation, and a large set of data for training the various system components we have just described. Without carefully architecting the system, the application would be inefficient and difficult to develop. In designing the MapPointS system, we have followed several design principles and software engineering considerations. In this section, we briefly describe these principles and considerations.

The first principle and consideration is *separation of interface and implementation*. Following this principle, we isolated components by hiding implementation details. Different components interact with each other through interfaces that have been well defined in advance. This allowed us to develop and test the system by refining components one by one. It also allowed us to hook MapPointS to different ASR engines without substantially changing the system.

The second principle and consideration is *separation of data and code*. MapPointS can be considered as a system whose design is driven by data and grammar. In the system design, we separated data from code and stored the data in the file system. The size of the data stored is huge since we need to maintain all the city names, place names, and their associated prior probabilities. By isolating the data from the code, we freely converted the system from one language to another by a mere change of the grammar, the place names, and the ASR engine for a new language.

The third principle and consideration is *separation of modalities*. We separated modalities of the speech input, text input, and the mouse input by representing their underlying semantic information in a common SML format. This allowed us to debug modalities one by one, and also allowed us to integrate more modalities in the future for possible system expansion by simply hooking the existing system to a new semantic parser.

The fourth principle and consideration is *full exploitation of detailed user feedback*. MapPointS provides detailed feedback to users in all steps that are carried out in processing the users' requests. In doing so, the users become able to know whether the system is listening to them and whether the ASR engine recognizes their requests correctly.

The final principle and consideration is *efficient design of the application grammar*. One of the significant problems of a large system like MapPointS is the creation of the specific application grammar, or grammar authoring. A good structured grammar can significantly reduce the effort in interpreting the results of speech recognition. In our implementation, we organized the grammar so that the semantic representation of the speech recognition results can be interpreted recursively.

5. Robust Processing Techniques for Speech-Centric HCI Systems

Robustness to acoustic environment, which allows speech recognition to achieve immunity to noise and channel distortion, is one key aspect of any speech-centric HCI system design considerations. For example, for the MiPad and MapPointS systems to be acceptable to the general public, it is desirable to remove the need for a close-talking microphone in capturing speech. The potential mobile application of MapPointS for navigation while traveling presents an even greater challenge to noise robustness. Although close-talking microphones pick up relatively little background noise and allow speech recognizers to achieve high accuracy for the MiPad-domain or MapPointS-domain tasks, it is found that users much prefer built-in microphones even if there is minor accuracy degradation. With the convenience of using built-in microphones, noise robustness becomes a key challenge to maintaining desirable speech recognition and understanding performance. Our recent work on speech processing aspects of speech-centric HCI systems has focused on this noise-robustness challenge in the framework of distributed speech recognition (DSR).

There has recently been a great deal of interest in standardizing DSR applications for a plain phone, PDA, or a smart phone where speech recognition is carried out at a remote server. To overcome bandwidth and infrastructure cost limitations, one possibility is to use a standard codec on the device to transmit the speech to the server where it is subsequently decompressed and recognized. However, since speech recognizers only need some features of the speech signal (e.g., Mel-cepstrum), the bandwidth can be further saved by transmitting only these features. Our recent work on noise robustness has been concentrated on the Aurora2 and 3 tasks [8, 15], an effort to standardize a DSR front-end that addresses the issues surrounding robustness to noise.

In DSR applications, it is easier to update software on the server because one cannot assume that the client is always running the latest version of the algorithm. With this consideration in mind, while designing noise-robust algorithms, we strive to make the algorithms front-end agnostic. That is, the algorithms should make no assumptions on the structure and processing of the front end and merely try to undo whatever acoustic corruption that has been shown during training. This consideration also favors noise-robust approaches in the feature rather than in the model domain.

We have developed several high-performance speech feature enhancement algorithms on the Aurora2 and 3 tasks and on other Microsoft internal tasks with much larger vocabularies. One most effective algorithm is called SPLICE, short for Stereo-based Piecewise Linear Compensation for Environments [3–5]. In a DSR system, the SPLICE may be applied either within the front end on the client device, or on the server, or on both with collaboration. Certainly a server side implementation has some advantages as computational complexity and memory requirements become less of an issue and continuing improvements can be made to benefit even devices already deployed in the field. Another useful property of SPLICE in the serve implementation is that new noise conditions can be added as they are identified by the server. This can make SPLICE quickly adapt to any new acoustic environment with minimum additional resource.

6. Summary and Discussion

Recent progress in signal processing and speech recognition technologies has created a promising direction for speech-centric multimodal HCI research. These HCI modalities include speech, vision (e.g., gesture), pen, mouse, keyboard, screen display, and other GUI elements. The speech-centric perspective for HCI advocated in this paper is based on the recognition that speech is a necessary modality to enable a pervasive and consistent user interaction with computers across a full range of devices—large or small, fixed or mobile, and that speech has the potential to provide a natural user interaction model. However, the ambiguity of spoken language, the memory burden of using speech as output modality on the user, and the limitations of current speech technology have prevented speech from becoming the choice of mainstream interface. Multimodality is capable of dramatically enhancing the usability of speech interface because GUI and speech have complementary strengths. Multimodal access will enable users to interact with an application in a variety of ways—including input with speech, keyboard, mouse and/or pen, and output with graphical display, plain text, motion video, audio, and/or synthesized speech.

Two prototype systems, MiPad and MapPointS, developed at Microsoft Research take the speech-centric perspective in their design. They fully exploit the efficiency of the speech input, while using other modalities to enhance the interaction and to overcome imperfection of the speech recognition technology. This paper provides a detailed account for the design of the MapPointS system. The system adds the "Speech" modality into the existing Microsoft product of MapPoint, which provides a comprehensive location-based database such as maps, routes, driving directions, and proximity searches. MapPoint also provides an extensive set of mapping-related content, such as business listings, points-of-interest, and other types of data that can be used within applications. In particular, it is equipped with highly accurate address finding and geo-coding capabilities in North America, and contains finely tuned driving direction algorithms using blended information from best-in-class data sources covering 6.7 million miles of roads in the United States. Loaded with the speech functionality, the value of MapPointS to the users is the quick, convenient, and accurate location-based information when they plan a long-distance trip, want to find their way around an unfamiliar town or try to find the closest post office, bank, gas station, or ATM in any town in North American. The MapPointS system has implemented a subset of the desired functionalities provided by MapPoint, limited mainly by the complexity of the grammar (used for semantic parsing), which defines what kind of queries the users can make verbally, possibly in conjunction with the other input modalities such as the mouse click and keyboard input.

We in this paper provided an overview of the MapPointS system architecture and its major functional components. We also presented several key software design engineering principles and considerations in developing MapPointS. One useful lesson we learned in developing MapPointS is the importance of user or environmental modeling, where the user-specific information and the user's interaction history with the system are exploited to beneficially adapt the LM. The drastically reduced perplexity of the LM not only improves speech recognition performance, but more significantly enhances semantic parsing (understanding) which acts on all types of input modalities, speech or otherwise. Some quantitative results we presented in Table 4 substantiated this conclusion.

Our current work is to apply the lessons learned from the MapPointS case study, user modeling in particular, as presented in detail in this paper to other speech-centric HCI tasks. For the extension of the prototype MapPointS system, we perceive the following future work:

- Port the system into mobile devices such as Pocket PC.
- Incorporate GPS information into the existing MapPointS functionality.
- Include new system functionalities such as direct address searching through speech.
- Improve the dialog system component in order to provide the speech response (instead of only the GUI response as is now), and to resolve confusability using speech interaction.

Acknowledgments

This invited paper is an expanded version of the keynote speech given by the first author at the IEEE Fifth Workshop on Multimedia Signal Processing in St. Thomas, US Virgin Islands on December 11, 2002. The authors wish to thank the organizing committee of the 2002 IEEE Workshop on Multimedia Signal Processing for the invitation to present the content of this paper at the workshop, and for the invitation to write this full-length paper. They also thank the members of the entire speech technology group at Microsoft Research in Redmond, who contributed many different components of the system described in this paper.

References

1. L. Comerford, D. Frank, P. Gopalakrishnan, R. Gopinath, and J. Sedivy, "The IBM Personal Speech Assistant," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, Salt Lake City, UT, May 2001.
2. L. Deng, K. Wang, A. Acero, H. Hon, J. Droppo, C. Boulis, Y. Wang, D. Jacoby, M. Mahajan, C. Chelba, and X.D. Huang. "Distributed Speech Processing in MiPad's Multimodal User Interface," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, 2002, pp. 605–619.
3. L. Deng, A. Acero, M. Plumpe, and X.D. Huang. "Large-Vocabulary Speech Recognition Under Adverse Acoustic Environments," in *Proc-ICSLP-2000*, Beijing, China, Oct. 2000, vol. 3, p. 806–809.
4. L. Deng, A. Acero, L. Jiang, J. Droppo, and XD Huang. "High-Performance Robust Speech Recognition Using Stereo Training Data," in *Proc. ICASSP-2000*, vol. I, Salt Lake City, Utah, April 2001, pp. 301–304.
5. J. Droppo, L. Deng, and A. Acero, "Evaluation of SPLICE on the Aurora2 and 3 Databases," in *Proc. ICSLP-2002*, Denver, CO., 2002.
6. S. Dusan, G. Gadbois, and J. Flanagan, "Multimodal Interaction on PDA's Integrating Speech and Pen Inputs," in *Proc. Eurospeech-2003*, Geneva, Switzerland, Sept. 2003.
7. R. Hamburgen, D. Wallach, M. Viredaz, L. Brakmo, C. Waldspurger, J. Bartlett, T. Mann, and K. Farkas, "Itsy: Stretch the Bounds of Mobile Computing," in *IEEE Computer*, April 2001, pp. 28–36.
8. H.G. Hirsch and D. Pearce, "The AURORA Experimental Framework for the Performance Evaluations of Speech Recognition Systems Under Noisy Conditions," in *Proc. ISCA*

ITRW-ASR2000 Automatic Speech Recognition: Challenges for the Next Millennium, Paris, France, Sept. 2000.

9. M. Johnston et al., "MATCH: An architecture for Multimodal Dialogue Systems," in *Proc. ACL-2002*, Philadelphia, PA, 2002.
10. J. Lai, (Ed.), "Conversational Interfaces: Special section," Communications of the ACM, vol. 43, no. 9, 2000.
11. A. Nefian, L. Liang, X. Pi, L. Xiaoxiang, C. Mao, and K. Murphy, "Dynamic Bayesian Networks for Audio-Visual Speech Recognition," EURASIP Journal of Applied Signal Processing, vol. 11, 2002, pp. 1–15.
12. C. Neti, G. Iyengar, G. Potamianos, A. Senior, and B. Maison. "Perceptual Interfaces for Information Interaction: Joint Processing of Audio and Visual Information for Human-Computer Interaction," in *Proc. ICSLP-2000*, Beijing, vol. 1, 2000, pp. 11–14.
13. S. Oviatt, "Breaking the Robustness Barrier: Recent Progress on the Design of Robust Multimodal Systems," in M. Zelkowitz, Advances in Computers (Ed.), Academic Press, vol. 56, 2002, pp. 305–341.
14. R. Rose, S. Parthasarathy, B. Gajic, A. Rosenberg, and S. Narayanan, "On the Implementation of ASR Algorithm for Hand-held Wireless Mobile Devices," in *Proc. ICASSP-2001*, Salt Lake City, Utah, vol. I, 2001.
15. J. Segura, A. Torre, M. Benitez, and A. Peinado, "Model-Based Compensation of the Additive Noise for Continuous Speech Recognition: Experiments using the AURORA2 Database and Tasks," *Proc. Eurospeech-2001*, Aalborg, Denmark, 2001.
16. R. Shama, V. Pavlovic, and T. Huang, "Toward Multimodal Human-Computer Interface," in *Proc. IEEE*, vol. 86, no. 5, 1998, pp. 853–869.
17. K. Wang, "Implementation of a Multimodal Dialog System Using Extended Markup Language," in *Proc. ICSLP-2000*, Beijing, China, 2000.

Au: Pls. provide Bios and Photo.