**MultiModal First Person Sensing**

*Qifan Hu ……more*

Carnegie Mellon University

5000 Forbes Ave, Pittsburgh, PA, USA 15213

<qhu@andrew.cmu.edu>

**ABSTRACT**

This report gives an overview of the MultiModal First Person Sensing (MMFPS) system, which is designed to create ambient intelligence in information-rich contexts. The multimodality nature of this system takes, analyzes real-time video and audio from a first person’s perspective, and afterwards, infers the real demands of the user, and provides seamless interaction between the user and the environment. Work towards the thoughts, design, development of the MMFPS system will be discussed.

1. **INTRODUCTION**
   1. **MMFPS System Overview**

There are three major components within the MMFPS system, namely the speech recognition, vision recognition, and machine learning integration. The speech recognition and the vision recognition components operate in a parallel fashion, and then the machine learning component will generate system results with the highest posteriori probability based on recognition accuracy and prior knowledge. The system architecture of the whole system is shown in Figure 1. A rounded box in the architecture indicates a function module, where recognizers (both speech and vision) for similar purposes are grouped together.

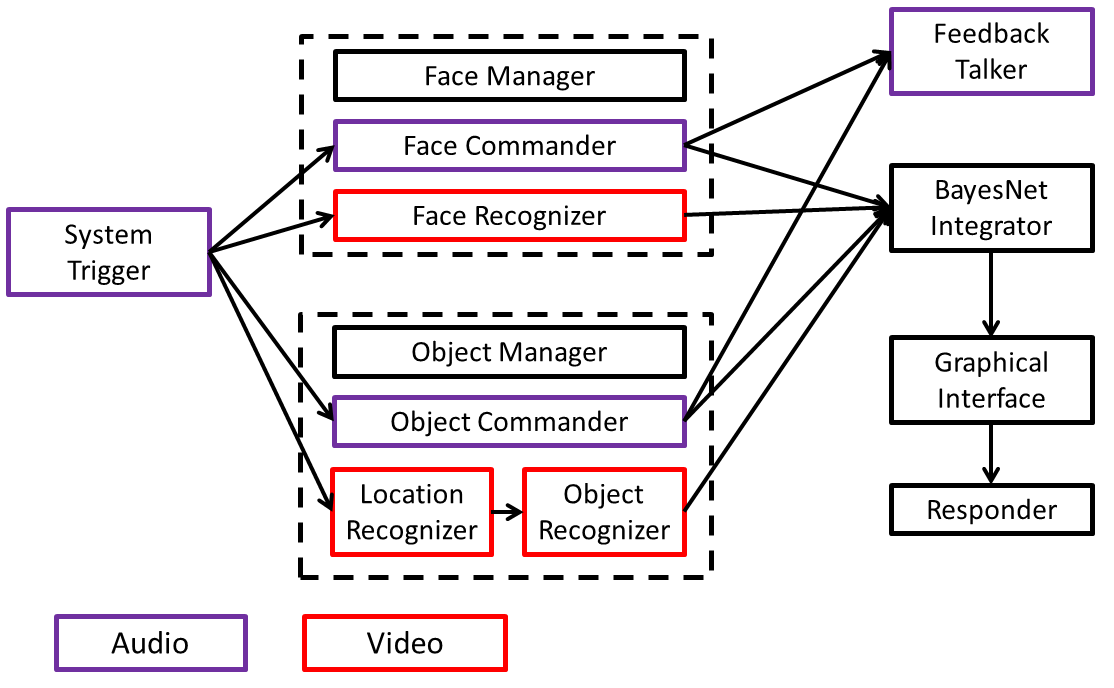


Figure 1:MMFPS System Architecture*. Purple boxes indicate speech recognizer components and red boxes indicate vision recognition components. To prevent ambiguity, “Commander” means “speech recognizer”, and “Recognizer” means “vision recognizer” in this paper.*

Current components in the MMFPS system include Speech Recognition (PocketSphinx [1]), Object Recognition (Exemplar-SVM [2]), Face Recognition [3], Location Recognition [4] and Bayesian Network Classifiers [5].

* 1. **Module Structure**

In this architecture, a functional module has one specific purpose, and a standard module is composed of four parts, the manager, the commander (speech recognizer), the recognizer (visual recognizer) and the machine learning integrator, as shown in Figure 2.

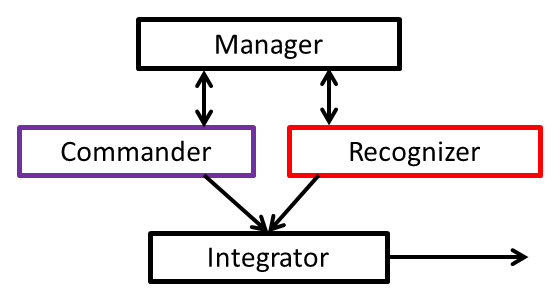


Figure 2:Module Structure. *There are four major components, while each box represents a class of recognizers or commanders. For example, both retail object recognizer and office object recognizer can be put in the recognizer box.*

The manager in the diagram is a singleton header class, which manages data for all the components in this module, including add, delete, and update entries.

The commanders, which recognize the actual user commands, are activated simultaneously once the system is triggered, and they are also responsible for parsing the semantic meanings of the recognized words. Each individual commander is equipped with a unique language model and a dictionary, as different commanders are supposed to recognize commands for different scenarios. In normal speech dialog systems, the user has to explicitly indicate which scenario it is, while in our system, all commanders are triggered together, and the scenario is dynamically determined by the recognition results.

This is s standardized functional module.

* 1. **Data Mangement**

dasdsadas

* 1. **Speech Dialog System Overview**

This report only includes details of the speech dialog system, and the architecture of this subsystem is shown in Figure 3. Generally, there are four major components in this speech dialog system, namely the Trigger, Dialog Manger, Commander (functional recognizer), and Feedback Talker.

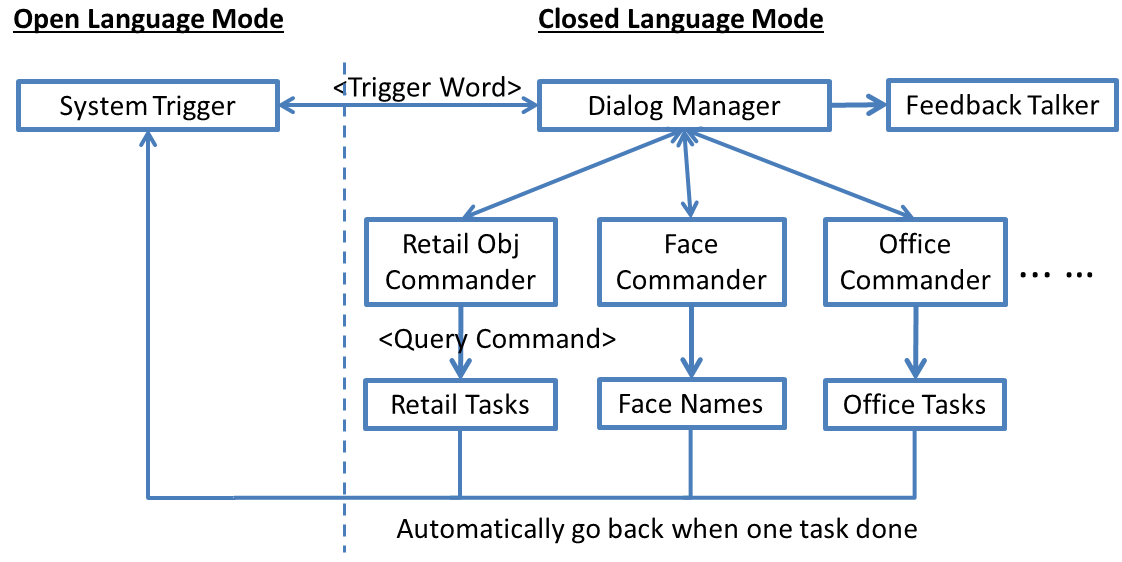


Figure 3:MMFPS Speech Dialog System Architecture*. This is a general representation of all the components, providing scalability and extensibility. While in practice, “Office Commander” and it relevant parts are not implemented. To prevent ambiguity, “Commander” means “speech recognizer”, and “Recognizer” means “vision recognizer” in this report.*

Due to the portability nature of the MMFPS system, the user is supposed to wear our device all the time. In this case, it becomes annoying if the system recognizing stuff that it is not supposed to recognize. For example, the user is talking with a friend, and the system should keep deactivated until the user requires assistance from it. To tackle this problem, we introduced the concept of “Open Language Mode”, as opposed to “Closed Language Mode”. In this open mode, the system trigger keeps detecting whether the user says the trigger word or not, preventing the actual commanders from recognizing noises.

An explanation of the “Open Language Mode”. The language corpus file only includes very limited number of words, instead of meaning sentences. The speech recognizer will recognize noises as random combinations of the words defined. A unique trigger word is inserted into the corpus, and it will trigger the overall system once it is detected. In the experiments session, the selection of the trigger word will be discussed.

Dialog Manager manages the flow of the whole system, which nodes to activate and deactivate. It monitors the results of all commanders, and determines which result reflects the actual meaning of the user, and then sends to the feedback talker for confirmation. Every time a command is issued, the system will go back to the open language mode, and keep deactivated until the user triggers it again. In another case, if the user realized the results recognized are not correct, he/she can repeat the process.

* 1. **Language Analysis**

In the first version of the speech recognition system, grammar models have been adopted to provide a formal representation of possible user commands. This technique is proven to be successful, with reasonable accuracy. However, one deficiency with grammar models is that, the user has to follow some specific patterns, which is, obviously, not feasible in real environments. To overcome this problem, statistical language models were adopted afterwards.

There are two components in a desirable semantic command, the verb and the noun. For example, “recognize this person” is clear to the system, what the user wants from it. Given a sentence, it is relatively straightforward to check which noun appears in the command, while for verbs, it becomes much more difficult. In our design, we used a simple machine learning classifier based on the occurrences of predefined keywords. For example, in the sentence “how do people like this book”, the keyword “like” indicates that the user wants to know the review about this book. In cases when more than one verb keyword are recognized, past queries are referred to determine the actual command.

* 1. **Adaptation**

We have defined four types of adaptation, as indicated in table 1 [6]. Among them, user-controlled user adaptation and system-controlled situation adaptation are adopted in this speech system. To put it easily, system-controlled situation adaptation means that, the system determines which scenario the user is using the system in, without any explicit indication from the users. Similarly, user-controlled user adaptation means that, the user explicit indicates the system his personal habits, allowing solid user adaptivity for the system.

|  |  |  |
| --- | --- | --- |
|  | ***Situation attributes*** | ***User attributes*** |
| ***User control*** | User-controller situation adaptation | User-controlled user adaptation |
| ***System control*** | System-controlled situation adaptation | System-controlled user adaptation |

Table1: Four Types of Adaption.

To be more specific, a full-size vocabulary is fed to all speech recognizers, while language models differs between each other. Whenever the user issues a command, all recognizers try to understand the command, and, in most cases, only one of the recognizers is able to get reasonable results. The system figures out the situation of the user based on this result, so this is called system-controlled situation adaption. For user-controlled user adaptation, the user explicitly commands the speech engine to add or delete one data entry to the system repository, so that other recognizers can realize the update in the database to adapt themselves to the user. This feature makes more sense in the multimodal context than in the speech-alone context.

* 1. **Development**

The overall system is built upon the work of Robot Operating System (ROS) [7], which is a free and open-source platform for advanced systems. In the ROS context, each recognizer is a standalone node, which keeps a finite state machine to keep track of the internal state. The communication between nodes is implemented in a publishing and subscription fashion. When a node wants to deliver some messages to other nodes, it publishes the message to the common space, and the target node will take it as input. Each node can has multiple publishers and subscribers.

In this project, a ROS wrapper was designed for an actual speech recognition engine, which provides full functionalities for each recognizer self and for its communication with other recognizers. The wrapper template is directly available for future usage, and it is shown in Figure 4.

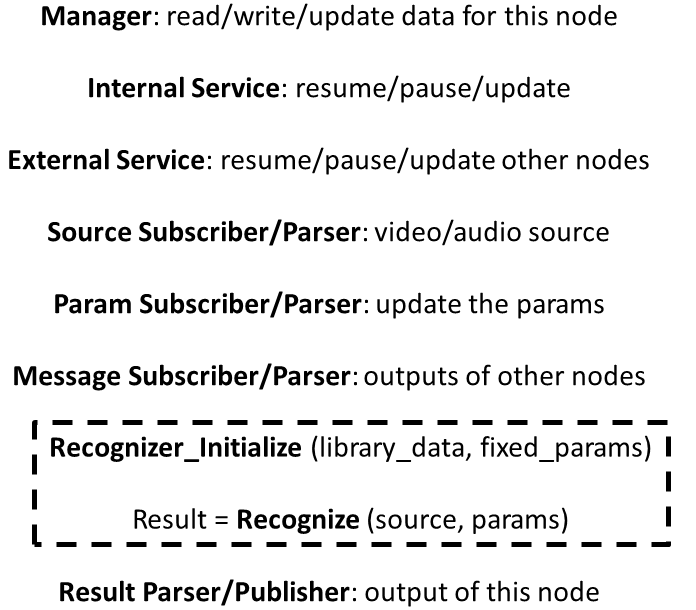


Figure 4: Generic ROS Wrapper Template*. It provides interfaces between this recognizer node and all other nodes, and therefore, the overall system.*

In the wrapper template, Manager manages the data for the recognizer; Internal services provide interfaces for other nodes to control this node self, and external services controls other nodes instead. Each node should have three kinds of subscribers, namely source subscriber, which takes real-time audio or video data; parameter subscriber which dynamically changes the parameters for the recognizer; message subscriber which monitors results of other nodes. Every time this node is activated, it will call the recognizer wrapped around, parse and publish the recognition result.

The speech dialog system is developed under Linux platform using PocketSphinx[1]. PocketSphinx provides a free, real-time continuous speech recognition solution for embedded devices, which is light weighted and fast. The original choice was Sphinx4, which is designed in Java. It provides decent performance and relatively new features, but limited functions under embedded environment, and poor compatibility against other system components. There are 16 major parameters in the PocketSphinx system to adjust the behavior and performance of the recognition engine. Table 2 lists some of the parameters. The performance experiments of the speech recognition engine will be based on values of these parameters.

|  |  |  |
| --- | --- | --- |
| ***Parameter Name*** | ***Value Type*** | ***Description*** |
| ~hmm | String | HMM directory |
| ~latdir | String | Lattice directory |
| ~lm | String | LM file |
| ~dict | String | Dictionary file |
| ~fsg\_model | Object | Finite state grammar object |
| ~fwdflat | Boolean | Flat lexicon search |
| ~bestpath | Boolean | Graph search |
| ~lattice | Object | Word lattice |
| ~maxhmmpf | Integer | Maximum HMMs per frame |
| ~maxwpf | Integer | Maximum words per frame |
| ~dsratio | Integer | Frame down sampling ratio |
| ~decoder | Object | Decoder object |
| ~configured | Boolean | Finalize configuration |

Table 2: Major Parameters in PocketSphinx

1. **EXPERIMENTAL RESULTS**
   1. **Open Language Mode Experiments**

As described above in the speech dialog system overview session, the open language mode is specifically designed to prevent the system from getting triggered by random noises. There are limited number of words in the corpus file, the language model and the dictionary. Once a specific trigger word is detected, then the system gets activated.

Based on the experiments, a single short word with limited characters has very poor performance, such as “hi”, “some”, “need”, etc. There are two options to replace this simple trigger word. The first option is to use a phrase combined of two or three simple words, such as “hello world”, “are you there”, etc., and the other option is to use a long yet unique word, such as “olympics”, “candidate”, etc. For option one, when the trigger phrases are composed of two simple words, then it still has a high False Positive rate, since either one can be trigger easily. If the phrase is longer than two words, then it becomes difficult for the user to memorize and use. Finally option two is adopted, and our trigger word choice is “Intelligentsia”, which becomes the name of the system. Experiment results based on “Intelligentsia” is listed in Table 3.

|  |  |  |  |
| --- | --- | --- | --- |
| ***True Positive*** | ***False Positive*** | ***True Negative*** | ***False Negative*** |
| 68.25% | 0% | 100% | 31.75% |

Table 3: Experiment Results in Open Language Mode

* 1. **Closed Language Mode Experiments**

The results are mentioned in the detection accuracy of key words, as explained the language analysis session. For example, the user says “Could you please give me reviews about this book?”, in this case, if both “review” and “book” are recognized, then we count it as correct, other situations count as incorrect. By adjusting the recognition engine parameters, we can get the results listed in Table 4. Generally, for larger language models, we increase the values of “maxhmmpf” and “maxwpf”, vice versa.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **~fwdflat** | **~bestpath** | **~dsratio** | **~maxhmmpf** | **~maxwpf** | **Accuracy** |
| **Set 1** | False | False | 1 | 1000 | 10 | 47.2% |
| **Set 2** | False | False | 2 | 3000 | 20 | 38.9% |
| **Set 3** | True | True | 1 | 1000 | 10 | 44.4% |
| **Set 4** | True | True | 2 | 3000 | 20 | 63.9% |

Table 4: Experiment Results in Closed Language Mode of One Commander

1. **CONCLUSION AND FUTURE WORK**

The architecture of the MMFPS system was changed couple of times before what it looks like today. The speech dialog system, however, presents a relatively stable architecture through experiments. Personally, I guess one of the reasons is that, systems built around speech recognition have been pretty mature, and a general architecture has been proven successful. In terms of recognition accuracy, it is relatively hard to evaluate, analyze and repeat the performance since there are many variables and uncertainties in a speech recognition system. Future work includes continued development of the current system, accuracy improvements and more experiments.

The Multi Modal First Person Sensing (MMFPS) Project is sponsored by the Intel Science and Technology Center on Embedded Computing (ISTC-EC).

1. **REFERENCES**
2. Huggins-Daines, D. and Kumar, M. and Chan, A. and Black, A.W. and Ravishankar, M. and Rudnicky, A.I., *PocketSphinx: A free, real-time continuous speech recognition system for hand-held device*, Acoustics, Speech and Signal Processing, vol.1, 2006.
3. Malisiewicz, T. and Gupta, A. and Efros, A.A., *Ensemble of exemplar-svms for object detection and beyond*, Computer Vision (ICCV), 2011 IEEE International Conference on, pp.89-96, 2011.
4. Heisele B. and Serre, T. and Poggio, T., *A component-based framework for face detection and identification*, International Journal of Computer Vision, vol.74, no.6, pp.161-181, Springer, 2007.
5. H. S. Park and Y. Sheikh, *3D reconstruction of a smoothed articulated trajectory from a monocular image sequence*, International Conference on Computer Vision (ICCV), 2011.
6. Friedman, N. and Geiger, D. and Goldszmidt, M., Bayesian network classifier, International Journal of Machine Learning, vol.29, no.2, pp.131-163, Springer, 1997.
7. Niklfeld, G. and Finan, R. and Pucher, M., *Architecture for adaptive multimodal dialog systems based on VoiceXML*, Proceedings of EuroSpeech, 2001.
8. Quigley, M. and Gerkey, B. and Conley, K. and Faust, J. and Foote, T. and Leibs, J. and Berger, E. and Wheeler, R. and Ng, A., *ROS: an open-source Robot Operating System*, ICRA workshop on open source software, vol.3, no.3.2, 2009.