2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN)
4-7 March 2018
Las Vegas, Nevada, USA

A Robust User Interface for IoT using Context-aware Bayesian Fusion

Jian Wu, Reese Grimsley and Roozbeh Jafari

***Abstract*— As the Internet of Things (IoT) continues to expand into our daily lives, consumers are finding a growing catalogue of smart devices to boost the intelligence of their homes. Currently, the user must manage a proprietary user interface (UI) for each device, and each application comes with its own UI, creating a cumbersome app environment. Clearly, a single UI that can control all of these devices would be preferable. This interface should be accessible using forms of communication that feel natural, for example, speech, body language, and facial expressions, to name a few. In this paper, we propose a framework for multimodal UI using a flexible, slotted command ontology and decision-level Bayesian fusion. Our case study explores command recognition for device control with a wearable system accessed via speech and gestures, using a wrist-mounted inertial measurement unit (IMU) for hand gesture recognition. We achieve an accuracy of 94.82% on a set of 17 commands.**

I. INTRODUCTION

In the last few years, the consumer market for intelligent home devices has grown rapidly. Internet-connected appliances allow users to augment the functionalities of their traditional counterparts in new and interesting ways. These devices are typically accessed via proprietary smart phone apps or physical interfaces. As these smart devices replace their regular versions, they develop a web of Internet-enabled gadgets, progressing towards the idea of a Smart Home. Systems such as Amazon Alexa and Google Home provide platforms for smart-device integration using speech queries. However, to achieve a more comprehensive and intuitive interface, we explore building system that utilizes multiple forms of natural human communication.

Most daily conversation, i.e. Human-Human Interaction (HHI), employs several communication methods, used jointly to convey thoughts more precisely. When we express our thoughts, we can do so most effectively by using a mixture of familiar expressions such as hand gestures, speech, eye gaze and posture. To facilitate Human-Computer Interaction (HCI) that feels intuitive, a combination of natural communication modes, such as those previously stated, should be utilized. In this way, the user can communicate their intentions more expressively and richly than with traditional computer input [1].

Jian Wu is with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77840, USA jian.wu@tamu.edu

Reese Grimsley is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77840, USA reesul@tamu.edu

Roozbeh Jafari is with the Department of Electrical and Computer Engineering, Computer Science and Engineering and Biomedical Engineering, Texas A&M University, TX 77840, USA rjafari@tamu.edu

Multimodal systems can offer superior robustness and flexibility compared to unimodal systems [2]. A particular modality may not be very useful in certain settings, for example, speech in a noisy room or gestures in a crowded area. In these scenarios, a unimodal system would be less accurate or even useless. In contrast, multimodal systems can improve robustness by using complementary or redundant input to disambiguate the users intent and increase certainty in the results. Such systems also allow the user to interact more flexibly, providing the option to select their preferred modalities. A study using a Wizard of Oz (WOz) approach [3] to evaluate HCI found that subjects used a combination of speech and gesture more often than individual modalities [4]. A similar WOz study for a speech and pen interface found 95% of participants preferred to interact multimodally, and that multimodal interaction facilitated faster task completion and fewer task-critical errors [5].

In addition, equivalent interactions in a given modality can differ drastically from person to person, creating a unique "fingerprint" for each user. For example, a person's accent may affect how their speech is recognized, and gesture models must often be tuned to match the user's physical characteristics. The UI should be customized to the user to provide consistent, accurate recognition of their actions. Furthermore, the UI should travel with the user from application to application, as opposed to the current norm in which the interface is application-specific. We envision that the UI must be abstracted out from the application, and remain with the user. A Unified interface can then enable interaction with a variety of applications.

In this paper, we introduce a framework for a multimodal user interface, fusing multiple modalities. We implement fusion using a Bayesian Network to recognize commands in a flexible, slotted structure. Our case study explores building commands for smart home devices, using cloud-based speech recognition and inertial measurement unit (IMU) based gesture recognition. The fusion takes into account the command history and gesture direction to fill in missing command parameters. We also introduce adaptive modality training to allow additional command customization. When a complete command has been recognized but unknown words or gestures are used, unfamiliar gestures can be assigned semantic meaning, and unfamiliar words can be added to the dictionary associated with each command function.

II. RELATED WORKS

The vast majority of research into gesture recognition relies on cameras to capture the user's movements [6], [7]. These vision-based recognition systems extract features from

978-1-5386-1109-8/18/$31.00 ©2018 IEEE

each graphical frame to classify the subjects movement [7], [8]. Although these systems have improved in recent years to reduce computational and economic costs, they require the user to be within the camera's field of view [9]. To facilitate ubiquitous smart device control, we envision a wearable system in order to reduce restrictions on the user. Thus, we have chosen to use inertial measurement units (IMUs), which are present in most mobile computers, for gesture recognition. This approach utilizes the acceleration and angular velocity of the system to classify the users motion. A system for IMU-based gesture recognition using Decision Trees and Support Vector Machines achieved up to 89% accuracy, tested on several different configurations of IMUs on the hand [10]. A study for recognizing American Sign Language (ASL) using an IMU and surface electromyography (sEMG) achieved 92% accuracy on 80 common ASL signs [11].

As the primary mode of communication for day-to-day conversation, speech is used by many multimodal HCI systems, often as the dominant modality [12]–[14]. Implementing offline automatic speech recognition (ASR) is computationally intensive, and its effectiveness can be compromised by dictionary size, noisy signals and so on [15]. However, many of these limitations can be avoided by using cloud-based ASR services such as Google Speech API, which can take advantage of extremely large amounts of training data to improve speaker-independent recognition on a large vocabulary of words, and can handle casual, conversational speech with noisy input [16].

Multimodal fusion is used to combine multiple, distinct input modalities into a single command or result. Feature-level fusion involves extracting a set of features from each input signal, and applying a classification algorithm to produce a result [17], [18]. Another popular approach fuses at the decision-level, and is the most common method used in HCI [19]. Individual modality recognizers classify the inputs received before the system attempts to fuse their results. These separate results require the system be trained in two phases: for the individual modalities and for the multimodal fusion, rather than single-phase training for the feature-level approach to fusion. However, modalities can be more easily added or removed with decision-level fusion, improving the system's extensibility [20]. We explore decision-level fusion in this paper, using a slotted command ontology, or frame-based approach. Semantic information is extracted from each input, which will be associated with a slot or frame; the slots are then combined to develop the resulting command [12]. A system for human-robot interaction achieved 97% accuracy on multimodal commands fusing cloud-based speech recognition and vision-based gestures [21].

III. METHODS

A. Proposed System Overview

The proposed multimodal user interface is shown in Fig. 1. There are three layers in our system: user input layer, customized application-agnostic user interface layer, and application layer. The user input layer includes three input modalities: hand gesture, voice, and location. In our paper, a

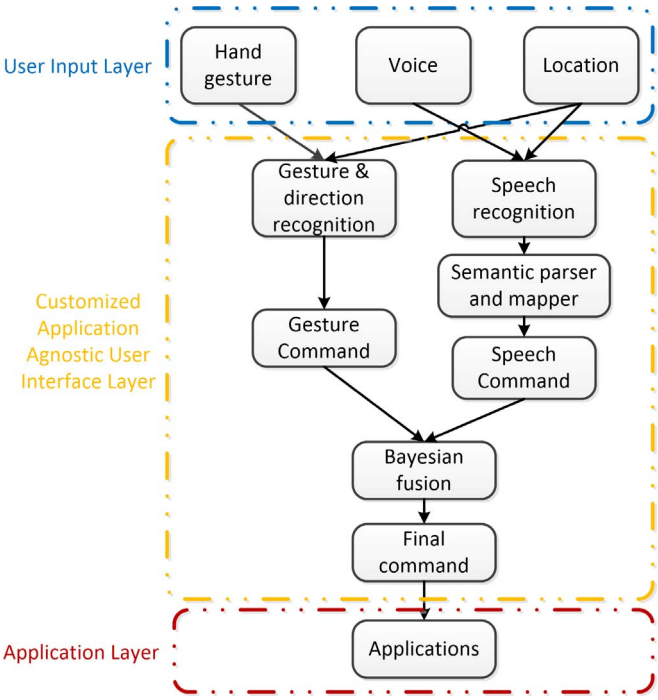

Fig. 1: Diagram of proposed system

9-axis, custom wrist-worn IMU is used to capture the hand gesture, and the microphone on a personal laptop is used to capture voice [11]. The second layer is our customized application-agnostic user interface layer, which translates the user input to a command for a certain application in the third application layer. The details of our user interface are introduced in the second layer. The gesture data from the IMU is used for gesture recognition and direction recognition. The direction recognition tells us which smart device the user intends to interact with, provided the user's location is known. The location information could be obtained by GPS sensors or indoor localization solutions, which we do not implement in our system and we assume we use off-the-shelf solutions. The gesture and direction recognition will provide the gesture modality's part of the command. The speech recognition module will translate speech into text, and a semantic parser and mapper performing natural language processing (NLP) will translate it into the speech command. Both the gesture and speech command will be represented in a slotted command structure proposed by this paper in III-E. Once both commands are known, a command completeness check engine will examine the command for completeness, i.e. all necessary slots filled. This engine will also provide cross-check function that completes the command of one modality using useful information from the other modality. If the command is complete from these two modalites, a context-aware Bayesian network will fuse the commands and determine the final command. Otherwise, the system will either predict a complete command or discard it. Additionally, our system is able to adaptively train the new speech and gesture modalities based on the mapping between gesture and speech commands.

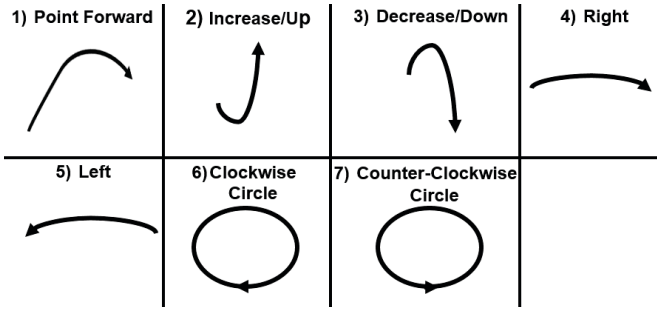

Fig. 2: Gesture shapes

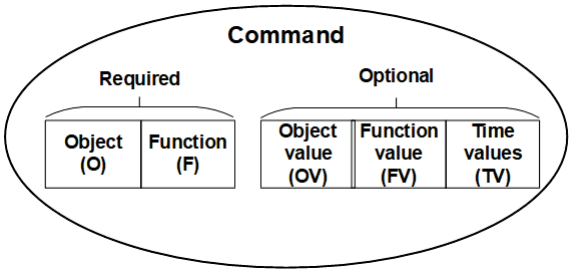

Fig. 3: Slotted command structure

B. Gesture Recognition

The raw 3D accelerometer and gyroscope data are used for hand gesture recognition. A real-time, online stream-matching algorithm called SPRING, an implementation of dynamic time warping, calculates the distance between the incoming stream and prerecorded templates for each gesture [22]. Once the distances between the incoming stream and all gesture templates are calculated, a gesture is positively classified if the distance falls below a predefined threshold. If the threshold is too large, many negative gesture instances will be classified as target gestures. If the threshold is too small, many positive gesture instances might not be recognized. In this paper, the thresholds are tuned experimentally for 8 hand gestures. Fig. 2 depicts the shape of each performed gesture. We envision our system as a wearable device, so battery power must be conserved, limiting how often the microphone can be on. The eighth gesture's purpose is to control the speech recognizer, done by tapping the device to open and close the microphone. This gesture is not associated with a semantic meaning and is not directly used in any commands.

C. Direction Recognition

The direction of the gesture can tell us which object the user is interacting with if the relative location of the user and devices are known. This paper does not focus on the mapping between the relative location and the gesture direction. We assume the user stays in the same relative location. The calibrated magnetometer reading is used for direction recognition using the same real-time DTW implementation [22]. In the training phase, the same gesture is performed in different directions, and templates containing only magnetometer readings are recorded for each of these directions. Once the gesture is recognized, the corresponding magnetometer readings in this period will be compared to all the directional templates for the recognized gesture. The direction is determined to be the same as the template that gives the smallest distance, given it is smaller than the threshold distance.

D. Speech Recognition and Semantic Parser

The microphone on a personal laptop is used for speech recognition. We use Google Speech API, a powerful cloud-based speech recognition API that converts speech to text. The output from speech recognition is parsed using the Stanford CoreNLP library in Java [23]. The parser separates the input string into an array of individual words, and tags each word with a part of speech (e.g. noun, verb). In this paper, we assume a homogeneous structure for speech input. We leave heterogeneous structures and implementation of a sophisticated mapper and parser to future work, as natural language processing is not a core component of this paper. The parser assigns each word from speech recognition to potential slots in the command structure, discussed in III-E, depending on the part-of-speech tag and adjacent words.

E. Command Structure

In UIs, many commands can be represented as a collection of semantic arguments. The user in the scenario of device control has access to several internet-enabled appliances, each with a unique set of functionalities. As illustrated in Fig. 3, a command is composed of a set of slots which may be required or optional depending on the command. In our case, we define the required slots to be the object (O), i.e. the device being used, and the function (F), i.e. the action taken by the device. For example, the command "Lamp increase brightness" uses (O) = "lamp" and (F) = "brightness".

Some commands need additional information to ensure that the users intention is correctly identified. Several slots, such as function value and time value, can be optionally specified. Generally, speech works best to supply this extra information. The Function Value (FV) provides additional information for the function. For "TV increase volume by 5", FV contains "increase" and "5" because these specify how the function should changes the device's state. When multiple instances of a device are in a room, the Object Value (OV) may be used to identify the object as a specific instance of the device. The Time Value (TV) slot is used whenever a user wants to specify how they want their command executed temporally. Given the command "Lamp turn off in 15 minutes", the TV slot will specify that the command needs to be delayed and the delay's duration. A basic command is represented as $C = \{O,F,\{OV,FV,TV\}\}$. We also define the gesture command, GC, and speech command, SC, as commands obtained from gesture and speech, respectively. There are represented as $GC = \{O_G,F_G,\{OV_G,FV_G,TV_G\}\}$ and $SC = \{O_S,F_S,\{OV_S,FV_S,TV_S\}\}$.

F. Completeness Check Engine and Cross-check

Once the gesture command and speech command are both obtained, a completeness check engine will check whether

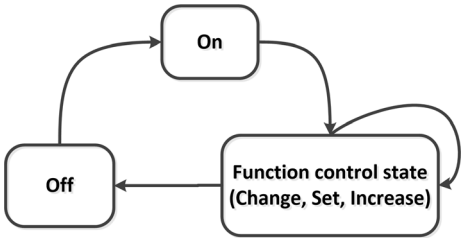

Fig. 4: State machine for reachability check

the command is complete. For example, for a command "lock the door", both gesture and speech recognize the "lock" functional element of a command. However, neither modality provides the object information ("door"), so the system will not consider the command complete. Only if the required slots are filled across all modalities will the commands be fused by the Bayesian network. To determine completeness, the truth value of this expression is checked: $(O_{\mathrm{G}} \parallel O_{\mathrm{S}}) \wedge (F_{\mathrm{G}} \parallel F_{\mathrm{S}})$. If true, the command is complete. Otherwise, the command is incomplete, and the system will predict a complete command based on device state reachability, discussed in III-G. The completeness engine will also perform a cross-check to complete the missing command element of one modality using the command from another modality. For example, in the command "turn on TV", if "TV" is not recognized by speech, and it is recognized by direction, then the engine will complete the speech command using gesture recognized command. Similarly, if the gesture direction could not identify the object in "turn on lamp", it can be completed with the speech recognized object. This simple cross-check will improve the accuracy for individual modality significantly.

G. Command Prediction Based on State Reachability

If the command is incomplete, our system will try to predict a command for the user based on state reachability for each device. Fig. 4 shows the state machine for each device. There are three states for a device: on, off and functional state. Therefore, the "turn on" command can only be followed by functional control command, and the "turn off" command can only be followed by "turn on" command. The function control command can either be reached by itself or the "turn on" command. In this step, we determine what the current state is, and traverse to all its neighboring reachable states to determine if the current command can be executed. For example, if an oven is in "off" state, and an incomplete command "change temperature" comes. The system will determine this command is not for oven since the "functional state" is not reachable by "off" state.

It is challenging to predict the command elements other than the object element. For example, if the incomplete command only has the object "TV", we cannot predict the function because we do not have enough information to predict what the user wants the TV to do. However, if an incomplete command is "turn off", it is possible for us to predict an object element for this command. If only the object is missing, the system will check the state reachablity for all devices and pick the candidate set. There might be several candidates, and the system will determine the most recent active device as the object. Notice that this approach only provides the user a good candidate and cannot guarantee the correctness of the prediction.

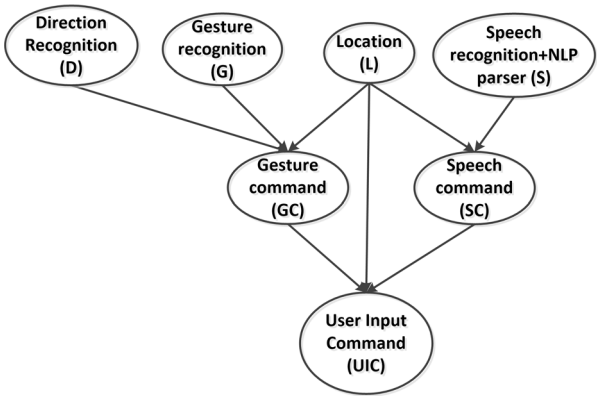

Fig. 5: Bayesian network for multimodal fusion

H. Context-aware Bayesian Fusion

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). There are four main features of Bayesian networks that make it ideal for our decision level fusion approach. First, it can handle missing data. Therefore, even if one modality or part of the command information is missing, the classifier is still able to work. Second, Bayesian networks allow one to learn about causal relationship. Third, it is an ideal representation for combining prior knowledge and data. The network structure is formed by the prior knowledge, and the probabilities are trained by the data. Fourth, it offers an efficient and principle approach to avoid overfitting data.

In Bayesian network, each node represents a random variable. If there is a directed link from node x to node y, x is defined as a parent of y. The node which does not have parent is called root node. To construct a Bayesian network, the network structure, the prior probability of root nodes and conditional probability of each node given their parents need to be given. Let $x = \{x_1,...,x_n\}$ represents the variable set, and PA_i denotes the parents of node x_i. The joint probability distribution of the Bayesian network is given by

$$P(x) = \sum_{i=1}^{n} P(x_i|PA_i) \tag{1}$$

The proposed Bayesian network is shown in Fig. 5. The location is important because we assume the user will only interact with certain devices in certain locations. For example, if the user is on his bed, he will not control the TV in the living room. The gesture command depends on a functional command recognized as the gesture and object command recognized as the gesture direction. It also depends on location of the user. The speech command depends

TABLE I: Command list

Command (ID)	Location	Command (ID)	Location
Turn on thermostat (1)	living room/bedroom	Turn off thermostat (2)	living room/bedroom
Turn on TV (3)	living room	Turn off TV (4)	living room
Increase TV volumn by X (5)	living room	Decrease TV volumn by X (6)	living room
Turn on lamp (7)	living room/bedroom/kitchen	Turn off lamp (8)	living room/bedroom/kitchen
Change color lamp (9)	living room/bedroom/kitchen	Increase brightness lamp (10)	living room/bedroom/kitchen
Decrease brightness lamp (11)	living room/bedroom/kitchen	Set oven temperature X (12)	kitchen
Set oven time by X (13)	kitchen	Lock door (14)	living room
Unlock door (15)	living room	Activate security (16)	living room/bedroom
Deactivate security (17)	living room/bedroom		

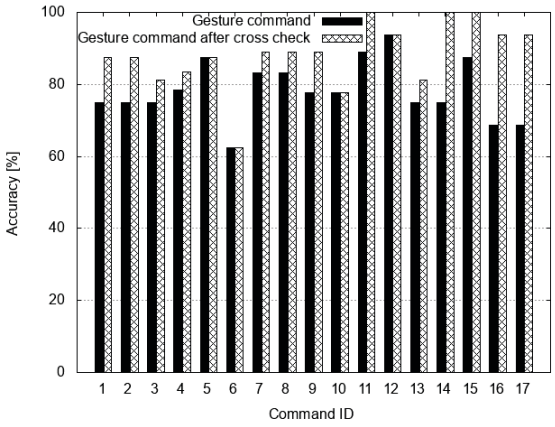

Fig. 6: Gesture command accuracy before and after cross-check

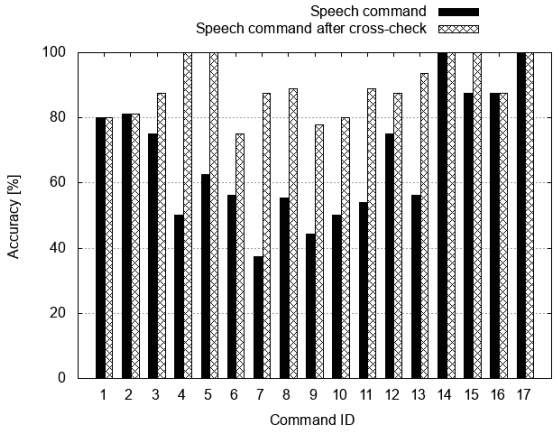

Fig. 7: Speech command accuracy before and after cross-check

on both speech recognition and the user's location. The final command depends on the gesture command, speech command, and the location. Thus, the system is context-aware by leveraging location and device state. The joint distribution of our proposed Bayesian network is given by

$$\begin{aligned} P(D,G,L,S,GC,SC,UIC) =& P(D)P(G)P(L)P(S) \\ & P(GC|D,G,L)P(SC|L,S) \\ & P(UIC|GC,SC,L) \end{aligned} \tag{2}$$

IV. Experimental Results

We test our system in a smart home environment, in which the user interacts with different objects at different locations. The commands are listed in Table I. There are 17 commands that include 6 objects and 3 locations in a smart home. For each command, both gesture and speech are repeated 15 times. The total number of instances is 255 in each modality. Some gestures are reused for different objects, and the total number of gestures in our test is 8 as introduced in III-B. The data is collected from one of the authors of this paper.

A. Command Recognition Accuracy after Cross-check

As discussed in Section III-F, if both modalities are present, a cross-check will be applied to complete missing commands elements in each modality. The cross-check completes missing command information of one modality using the corresponding recognized information from the other modality. Fig. 6 shows gesture command accuracy before and after crosschecking. As shown in the figure, the accuracy for most commands improves significantly. This is because we have a tight threshold for gesture and direction recognition, and either the gesture or direction could not be recognized as easily. In this case, the recognized speech command will help complete the gesture command; thus, the gesture command recognition accuracy is improved. Similarly, Fig. 7 shows the speech command accuracy before and after crosschecking. Some words have less consistent recognition than others. In addition, the NLP parser is not able to separate the command elements very well in some scenarios, especially when words for the object or function are not recognized at all. The information from the gesture command can help complete the speech command too. For example, 'TV' is difficult to recognize, and the NLP parser could not identify the object of command 'Turn off TV' due to differences with the assumed speech structure. By using object information from the gesture command, the accuracy of command 4 is improved significantly. We use a low-cost microphone and the API (Google Assistant SDK) offers worst recognition than on their own device (*e.g.* Google Home).

B. Command Recognition Accuracy after Bayesian Fusion

After the completeness check and cross-check, the gesture and speech command are both obtained, and they are fused using the Bayesian network. Fig. 8 shows the recognition

TABLE II: Average recognition accuracy

	Gesture only	Speech only	Gesture with cross-check	Speech with cross-check	Fused
Accuracy	78.43%	67.81%	88.62%	87.91%	94.82%

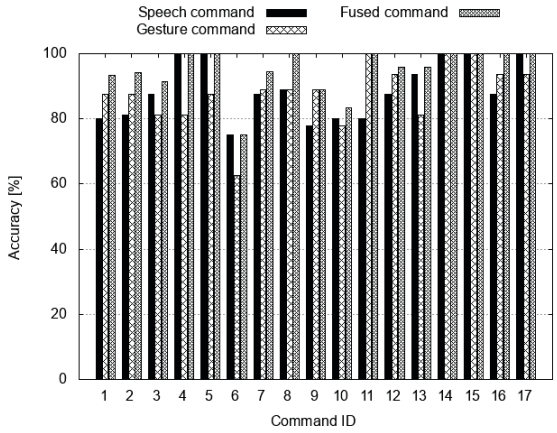

Fig. 8: Command accuracy for gesture, speech and fusion

accuracies for speech, gesture, and fusion for all commands. In the figure, command accuracies for speech and gesture are obtained after the cross-check. From the figure, the fused accuracy is greater than or equal to accuracy of the best performing modality for all commands.

Table II shows the average accuracy for gesture only, speech only, gesture with cross-check, speech with cross-check, and fusion. From the table, we observe the basic gesture and speech will achieve poor performance in command recognition when used individually. However, when they are used together, the performance will improve significantly. The final accuracy after Bayesian fusion can achieve 94.82%.

V. Conclusion

In this paper, we proposed a robust user interface for IoT using a slotted command ontology and decision-level Bayesian fusion. By using a slotted command, we can assign semantic meaning to results from any modality. This allows the UI to be tailored to different applications in which multimodal commands have a generic structure. Our system was evaluated in a smart home environment with 17 commands and the experiment results show an average accuracy of 94.82% for detecting the user commands and intent.

References

[1] B. Dumas, D. Lalanne, and S. Oviatt, "Multimodal interfaces: A survey of principles, models and frameworks," *Human machine interaction*, pp. 3–26, 2009.

[2] W. Lai and H. Huosheng, "Towards multimodal human-machine interface for hands-free control: A survey," *Techn. Ber., School of Computer Science & Electronic Engineering, University of Essex*, 2011.

[3] D. Salber and J. Coutaz, "Applying the wizard of oz technique to the study of multimodal systems," in *International Conference on Human-Computer Interaction*. Springer, 1993, pp. 219–230.

[4] M. Lee and M. Billinghurst, "A wizard of oz study for an ar multimodal interface," in *Proceedings of the 10th international conference on Multimodal interfaces*. ACM, 2008, pp. 249–256.

[5] S. Oviatt, "Multimodal interactive maps: Designing for human performance," *Human-computer interaction*, vol. 12, no. 1, pp. 93–129, 1997.

[6] A. Chaudhary, J. Raheja, K. Das, and S. Raheja, "A survey on hand gesture recognition in context of soft computing," *Advanced computing*, pp. 46–55, 2011.

[7] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[8] Q. Chen, N. D. Georganas, and E. M. Petriu, "Real-time vision-based hand gesture recognition using haar-like features," in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*. IEEE, 2007, pp. 1–6.

[9] Y. Ichikawa, S. Tashiro, H. Ito, and H. Hikawa, "Real time gesture recognition system with gesture spotting function," in *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. IEEE, 2016, pp. 1–7.

[10] A. Moschetti, L. Fiorini, D. Esposito, P. Dario, and F. Cavallo, "Recognition of daily gestures with wearable inertial rings and bracelets," *Sensors*, vol. 16, no. 8, p. 1341, 2016.

[11] J. Wu, L. Sun, and R. Jafari, "A wearable system for recognizing american sign language in real-time using imu and surface emg sensors," *IEEE journal of biomedical and health informatics*, vol. 20, no. 5, pp. 1281–1290, 2016.

[12] M. T. Vo and C. Wood, "Building an application framework for speech and pen input integration in multimodal learning interfaces," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 6. IEEE, 1996, pp. 3545–3548.

[13] H. Holzapfel, K. Nickel, and R. Stiefelhagen, "Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3d pointing gestures," in *Proceedings of the 6th international conference on Multimodal interfaces*. ACM, 2004, pp. 175–182.

[14] A. Corradini, M. Mehta, N. O. Bernsen, J. Martin, and S. Abrilian, "Multimodal input fusion in human-computer interaction," *NATO Science Series Sub Series III Computer and Systems Sciences*, vol. 198, p. 223, 2005.

[15] M. Forsberg, "Why is speech recognition difficult?" *Chalmers University of Technology*, 2003.

[16] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, "Google search by voice: A case study," in *Advances in Speech Recognition*. Springer, 2010, pp. 61–90.

[17] A. Jaimes and N. Sebe, "Multimodal human–computer interaction: A survey," *Computer vision and image understanding*, vol. 108, no. 1, pp. 116–134, 2007.

[18] A. Jagadeesan and K. Duraiswamy, "Secured cryptographic key generation from multimodal biometrics: feature level fusion of fingerprint and iris," *arXiv preprint arXiv:1003.1458*, 2010.

[19] R. Sharma, V. I. Pavlovic, and T. S. Huang, "Toward multimodal human-computer interface," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 853–869, 1998.

[20] S. Rossi, E. Leone, M. Fiore, A. Finzi, and F. Cutugno, "An extensible architecture for robust multimodal human-robot communication," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2208–2213.

[21] F. Cutugno, A. Finzi, M. Fiore, E. Leone, and S. Rossi, "Interacting with robots via speech and gestures, an integrated architecture." in *INTERSPEECH*, 2013, pp. 3727–3731.

[22] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007, pp. 1046–1055.

[23] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010