Rospeex: A Cloud Robotics Platform for Human-Robot Spoken Dialogues

Komei Sugiura and Koji Zettsu

***Abstract*— To build conversational robots, roboticists are required to have deep knowledge of both robotics and spoken dialogue systems. Although they can use existing cloud services that were built for other services, e.g., voice search, it will be difficult to share robotics-specific speech corpora obtained as server logs, because they will get buried in non-robotics-related logs. Building a cloud platform especially for the robotics community will benefit not only individual robot developers but also the robotics community since we can share the log corpus collected by it. This is challenging because we need to build a wide variety of functionalities ranging from a stable cloud platform to high-quality multilingual speech recognition and synthesis engines. In this paper, we propose "rospeex," which is a cloud robotics platform for multilingual spoken dialogues with robots. We analyze the logs we have collected by operating rospeex for more than a year. Our key contribution lies in building a cloud robotics platform and allowing the robotics community to use it without payment or authentication.**

I. Introduction

Speech interfaces have become widely used in such human-computer interactions as voice search, smartphone-based agents, and of course robots. A speech interface has advantages in various use cases when the user finds the convenience of interacting with the robots in a natural way by speech or when hands-free interactions are required. Although conversational robots have various applications in the service robot domain, building such robots is not simple.

Suppose a roboticist started building such a robot. Sooner or later he would find that more utterances are misrecognized than expected and that the robot voice sounds monotonous and unnatural. Currently, building conversational robots requires deep knowledge of both robotics and spoken dialogue systems. This prevents roboticists from spending more time on their own research topics for real applications.

In this study, our target use case is spoken dialogues with service robots that mainly work in domestic environments. An example use case is where a user asks a robot to fetch a plastic bottle from a table in RoboCup@Home [1]. Fig. 1 showcases such interaction where the speaker asks the robot ("Daia") to manipulate table-top objects.

In such tasks, robot-related middleware, including ROS [2] (Robot Operating System) or RT-Middleware [3], is usually used to reduce development cost. Most roboticists have been using speech recognition and speech synthesis toolkits provided with the above middleware [4], [5]. However, high-quality speech recognition is very difficult since stand-alone computers have CPU and memory limitations. Although they can use existing cloud-based APIs that were built for other services, e.g., voice search, it will be difficult to share robotics-specific speech corpora obtained as server logs, because they will get buried in non-robotics-related logs. In most cases, they are incompatible with the above middleware since their providers do not focus on robotic applications.

From the above background, we constructed a cloud robotics platform called "rospeex" that we have been operating for over a year. Unlike stand-alone approaches, roboticists do not need high-spec on-board processors for speech recognition and synthesis.

The following are our key contributions:

- We propose and construct a cloud platform for multilingual human-robot spoken dialogues. Any roboticist can use the proposed platform without payment or authentication. The platform is explained in Section III.
- We have been operating our proposed platform for more than a year. We analyzed more than ten thousand utterances collected as cloud server logs to investigate its feasibility and show the results in Section IV.

II. Related Work

One of the first studies on separating processing resources from the robot's body was conducted by Inaba as Remote-Brained Robotics [6]. Cloud Networked Robotics focuses on networked robotic services that cannot be satisfied by a stand-alone robotic system [7]. In 2010, Kuffner coined the term "cloud robotics" and described potential benefits when cloud infrastructures are fully utilized by robots [8]. Kehoe et al. surveyed cloud robotics and points out that big

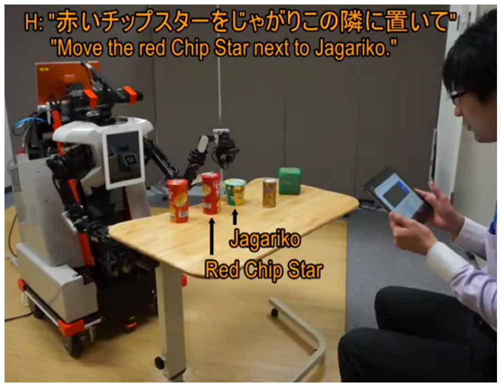
H: "赤いチップスターをじゃがりこの隣に置いて"
"Move the red Chip Star next to Jagariko."
Jagariko
Red Chip Star

Fig. 1. Sample dialogue using rospeex between a speaker and "Daia," our robot platform. Large vocabulary continuous speech recognition is possible. In other words, the grammar containing proper nouns such as "Chip Star" and "Jagariko" is not hand-coded.

Komei Sugiura and Koji Zettsu are with the National Institute of Information and Communications Technology, 3-5 Hikaridai, Seika, Soraku, Kyoto 619-0289, Japan. komei.sugiura@nict.go.jp

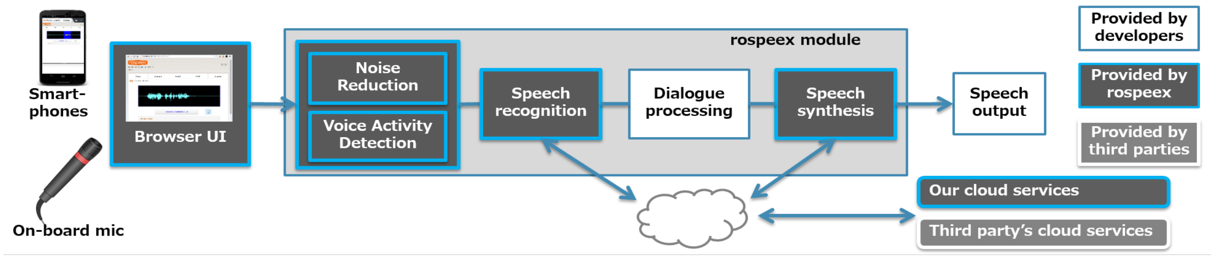

Fig. 2. Relationship of rospeex modules and cloud services. We provide rospeex modules and cloud services. Using third parties' APIs is simple since we also provide a function for switching cloud services. We assume that developers write code on dialogue-related functions.

data, cloud computing, collective robot learning, and human computation are four potential benefits of cloud robotics [9].

In cloud robotics studies, cloud services are used for object recognition, knowledge sharing, and machine learning [10]–[12]. Our study is closely related to these works; however, the difference is that our platform focuses on speech recognition and synthesis.

In robotics, stand-alone engines are usually used for speech recognition and synthesis. Speech processing toolkits such as HARK [5] and OpenHRI [4] utilize stand-alone engines Julius [13], Festival [14], or OpenJTalk [15]. However, the cloud approach has more benefits compared with the stand-alone approach. For example, having many uncommon proper nouns in the vocabulary (internal dictionary) is difficult due to resource limitations, so that they cannot be recognized. Moreover, building multilingual robots is time-consuming for developers. Technically speaking, It is technically true that these engines can be used for multi-lingual speech recognition, however this is actually difficult for non-expert of speech recognition. For example, most roboticists hesitate to switch language models and tune their system's performance. For multilingual spoken dialogues, our approach merely requires the developer to change one argument of a function such as "en" to "ja". Our study is also unique because the platform has been used by many roboticists worldwide, and a multilingual speech corpus was constructed as server logs.

On the other hand, some providers have started cloud services capable of speech recognition and synthesis mainly for smartphone-based services such as voice search and dialogue agents. Although the roboticists can use existing cloud services for voice search, e.g., it is difficult to share robotics-specific speech corpus obtained as server logs because they will get buried in non-robotics-related logs. Building a cloud platform especially for the robotics community will benefit not only individual robot developers but also the robotics community itself since we can share the log corpus collected by the platform.

Some providers have published cloud APIs (Application Programming Interfaces) that were originally developed for large-scale QA systems such as Watson [16] by IBM and "Shabette Concier" [17] by NTT Docomo. Although rospeex and these services have different foci, we can connect them and make an integrated robot system. By combining rospeex and these services, developers can make their robots highly interactive without building complicated knowledge databases.

III. Proposed Framework

Figure 2 shows the relationship of the rospeex module and cloud services [1] . We provide the browser user interface (UI), the rospeex modules (noise reduction, voice activity detection, and speech synthesis), and the rospeex cloud services. We assume that the developer writes code on dialogue-related functions including language understanding, dialogue management, and response generation. In this paper, we use the following two terms to distinguish two types of rospeex users.

- Developer: builds the robot's speech functionality to enable speech interactions
- Speaker: interacts with the robot. We do not consider complicated multi-party dialogues. In a strict sense, a speaker does not mean a listener (or an "addressee" [18]); we use "speaker" for readability.

A. Problem Statement

Suppose roboticists used existing cloud services (mainly for voice searches) to add dialogue-functionality to their robots. In such cloud servers, the logs of robotics-specific interactions get buried in other non-robotics-related logs (e.g., voice searches or translation), and the service providers would never consider extracting robotics-related knowledge due to cost-effectiveness.

Unlike merely using cloud services, our approach constructs our own cloud services and allows the roboticists to use them freely as an ROS (Robot Operating System) module so that we can store a huge amount of robotics-related logs in the cloud server. The logs can later be used to improve the language models for robotic tasks or shared with other roboticists as our community's shared asset. This explains why we provide our own cloud services.

The aim of this study is to construct a cloud platform for spoken dialogues and to analyze the robotics-related log corpus collected by our platform.

In this paper, we do not deal with acoustics including sound localization or noise reduction by a microphone array. Improving the quality of speech recognition or speech

[1]For demonstration videos and software, visit: http://rospeex.org/

synthesis is beyond the scope of this paper. We assume that dialogue processing is written by the developers.

B. Browser User Interface

The browser UI of rospeex is shown in Fig. 3. It is unique because it is written in HTML5 and runs multiple platforms such as Windows, Linux, and Android smartphones with a Mozilla Firefox browser.

This cross-platform interface gives the developer two options for selecting microphones. Here we do not consider microphone arrays.

- On-board microphone: The microphone is placed on the robot's body. The browser UI and the rospeex modules are running on the same PC.
- Smartphone microphone: The browser UI runs on a smartphone and is connected to the rospeex modules running on another PC. For reducing speech recognition errors, we recommend using smartphone microphones. We discuss this issue in Section V.

In the figure, the blue part shows the utterance segment that is automatically detected by the Voice Activity Detection (VAD). Having a graphical UI regarding speech input is important for actual service robots. Otherwise, it would be difficult for the speaker to imagine why the utterance was misrecognized. In human-robot interactions in a real (non-laboratory) environment, recognition errors are often caused by various reasons including microphone settings, noise, and VAD errors. However, the speaker often mistakenly believe that the speech recognizer caused the problem. By interactively showing the waveform in the UI, the speaker can distinguish potential causes, e.g., the voice was not loud enough, the microphone gain was not set appropriately, and so on. In the UI, the developer or speaker can switch off the VAD so that unnecessary utterances are not sent to the cloud server.

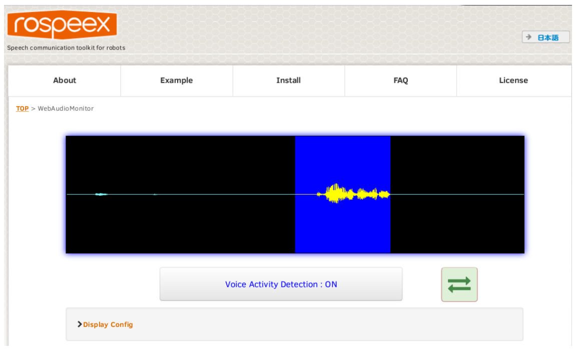

Fig. 3. Browser UI of rospeex. Blue shows utterance segment detected by Voice Activity Detection (VAD).

C. Noise Reduction and Voice Activity Detection

Noise is one of the biggest problems for applying speech recognition to robots. For example, the equivalent continuous noise level (Leq) is 75 [dB] in the standard task setup of RoboCup@Home competitions. In them, the distance between the speaker and the robot is about 1 [m], and the SNR is usually less than 5 [dB]. Noise reduction is needed.

For noise reduction, we use the standard noise reduction method with Minimum Mean Square Estimators (MMSE). A promising alternative is the particle-filter-based approach [19]. Although we showed that it works well for human-robot interactions in real environments [20], we did not adopt it in rospeex due to its high computation cost.

Another big issue specific for robotics is VAD. Most robots do not have a button to start speech recognition. Thus the beginning and the ending of the speaker's utterances have to be automatically detected by a VAD module.

In designing rospeex, noise reduction and VAD are not conducted on the server. Considering the network delay and since it is unreasonable to implement these functions on the server, we implemented them on the rospeex module as a ROS node.

D. Cloud-based Speech Recognition and Synthesis

The rospeex cloud servers for speech recognition and speech synthesis were originally developed for "VoiceTra," a speech-to-speech translation system [21] that is used by over one million users. Our cloud services support multilingual speech communication in Japanese, English, Chinese, and Korean; however, language detection is not supported.

For those who do not use ROS, we provide another way to utilize the rospeex cloud service. Our cloud servers allow developers to access it by a JSON interface to become compatible with C++, Python, JavaScript, etc. They can also obtain the recognition results by sending a command in JSON format (Fig. 4) to the following URL: `http://rospeex.ucri.jgn-x.jp/nauth_json/jsServices/VoiceTraSR`. "BUF" represents sound content (16kHz, 16bit, mono, and little endian) encoded with base64. The recognition result is sent back to the developer. For speech synthesis, the JSON interface is described in detail in [22].

Although our platform can be used with or without ROS, we recommend using with ROS because developers do not have to write the above JSON commands. With rospeex, they do not have to write much code for cloud-based speech processing. For example, less than ten lines are required to write a function for such simple dialogues as asking time.

For usability, we also provide a way to select other cloud services. The developers can switch cloud engines with very little effort and integrate rospeex with their software assets written for ROS. For example, integrating rospeex should be relatively simple with sound the localization functionality provided in HARK [5].

E. Non-Monologue HMM Speech Synthesis

Although natural conversations are desirable in human-robot interactions, most speech synthesis engines are not optimized for them. Therefore, the robot voices do not sound friendly and natural. If the intonation is not appropriate, the speakers might not realize that they are being asked a question.

```
{
   "method" : "recognize",
   "params" : (
      "ja",           // language
      {"audio": BUF, // base64-encoded wav
      "audioType": "audio/x-wav",
      "voiceType": "*"}
      )
}
```

Fig. 4. Speech recognition command in a JSON format for those who do not use ROS. Rospeex cloud services can be used both with and without ROS. With ROS, developers do not have to write the above JSON command.

For expressive speech synthesis, we built the Non-Monologue HMM-based Speech Synthesis method [22]. Rospeex is compatible with this method so that developers can use it for free. Unlike conventional methods, the robots using it can synthesize friendly and natural voices. In previous research, we evaluated it with the standard MOS metric and showed that its performance almost approached the theoretical upper limit. The details of the results are explained in [22].

IV. Experiments

A. Experimental Setup

We have been operating our proposed platform for more than a year. The rospeex cloud platform, which was launched on September 1, 2013, has obtained over 12,000 unique users, as of June 30, 2015. Here we define a unique user as an access to our platform from an IP address; multiple accesses from the same IP address on a single day are counted as a one unique user. Although we have been testing our proposed platform with our own robots including Daia (Fig. 1), we removed the accesses from our own institute for analyzing actual use cases.

Analyses were performed to investigate the feasibility of our proposed platform. In the next subsection, the cloud platform is evaluated in terms of processing speed to show the feasibility of our approach. This is because speed is critical issues when using cloud services instead of stand-alone systems. The cloud server had Intel X5690 CPUs (12 cores, 3.47GHz) and a 200-GB memory. The quantitative results of our speech synthesis method are shown in [22].

B. Result (1): Processing Speed by Cloud Servers

By analyzing the access logs from December 1, 2013 to May 31, 2014, we evaluated the cloud approach by investigating the processing time of speech recognition and synthesis. Although the waiting time also depends on network delay, it is beyond the scope of this paper since investigating it in many different environments/countries is unreasonable.

First, we show the processing time of speech recognition. Fig. 5 shows the histogram of the processing time. For most requests, it ranged from 200 to 2000 [msec], and the median was 961 [msec]. Here the term "request" is defined as a single processing request that is sent by a rospeex module to

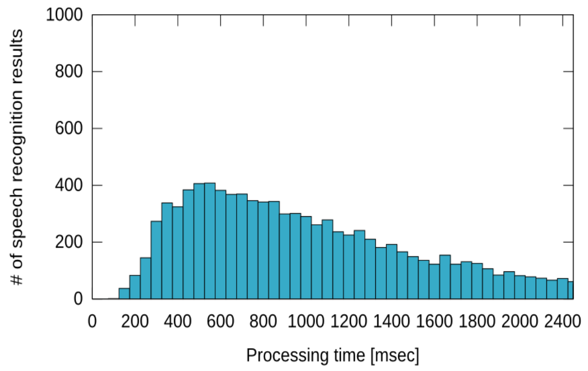

Fig. 5. Histogram of processing time for speech recognition.

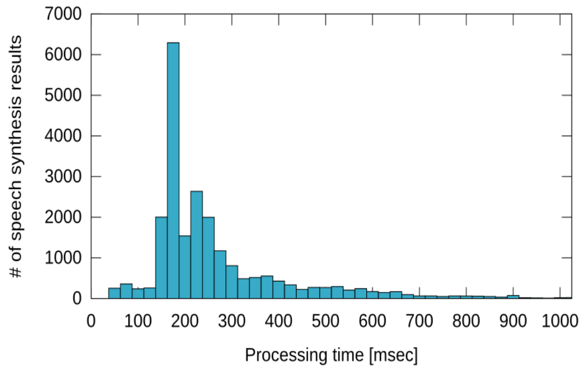

Fig. 6. Histogram of processing time for speech synthesis. Most requests were processed within 400 [msec].

the rospeex server. Although we omit the details due to space limitations, the processing time was almost proportional to the length of the input utterances.

In Fig. 5, some requests took more than 1000 [msec] to be processed. Such a long waiting time is harmful because it might deteriorate the interactive aspect of dialogues. We further discuss this issue in Section V.

Next, we show the processing time of speech synthesis. Fig. 6 shows a histogram of the processing time. From the figure, we can see most of the requests were processed within 400 [msec].

In Fig. 6, some speech synthesis requests took more than 1000 [msec] to be synthesized because these requests included multiple sentences that were copied from web sites. To reduce the waiting time, the developer should merely cut the multiple sentences into individual sentences before sending. This will considerably reduce the waiting time without much effort.

Table I summarizes the above results, and shows the number of requests, the processing times, and the real time factor (RTF). RTF is defined as follows:

$$\text{RTF} = \frac{T_p}{T_u}, \tag{1}$$

where T_p and T_u denote the processing time and the speech length. From Table I, the RTF in speech recognition was approximately 0.76. This means the processing time for an utterance was 76% of the original speech. For speech synthesis, the RTF was approximately 0.2, which indicates that the processing time was 1/5 of the synthesized speech length. This is considered practical for server-side processing. The above evidence indicates that the cloud platform responded within a reasonable amount of time.

C. Result (2): Analysis of Utterances

We analyzed the access logs between January 1, 2014 to November 28, 2014 and found that most developers generally applied rospeex to home robots. Thus, we categorized the utterances into categories regarding home robotics.

The total number of utterances was 44960; we filtered out the speech files that did not contain any sound. Table II categorizes the transcriptions whose frequency exceeded two. Here we define the categories as follows:

1) Basic communication: greetings and simple chatting
   e.g., "hello," "who are you?" "good-bye."
2) Question and answer: Simple information retrieval that does not require the history of the dialogues.
   e.g., "what time is it?" "show me my schedule," "give me today's weather forecasts."
3) Motion control: command utterances on locomotion or manipulation
   e.g., "stop," "turn right," "go to the bookshelf."
4) Controlling appliances: utterances to control home appliances
   e.g., "turn on the TV," "turn on the light."
5) Learning and recognition: command utterances to execute learning or recognition of sensory input
   e.g., "where are you?" "look at that."
6) Other command utterances: command utterances that are not categorized as Categories 3, 4, or 5.
   e.g., "raise your hands," "shut down the process."
7) Fragmented speech: VAD errors or utterances that are uncategorizable due to missing context.

From Table II, we can see that almost half of the utterances are categorized into Categories 1 and 2. The table indicates that the speakers rarely input motion control commands but they often input greetings. This can be explained by the fact that the speakers were probably the developers themselves who used rospeex in the test phase.

The table also indicates a practical strategy for building conversational robots. For example, it is effective to have standard cloud-based QA APIs in dialogue processing for solving Category 1 and 2 tasks. Natural spoken language understanding on Category 3 to 5 utterances should not be difficult, though building functions corresponding to them is time-consuming and difficult. In contrast, solving Category 7 tasks requires innovations in many aspects ranging from speech processing to deep analyses of situated dialogues.

Future work will improve the accuracy of VAD and speech recognition to reduce the utterances miscategorized into Category 7. It will also include grounded dialogues that utilize context information, such as the history of the dialogue and grounded information.

TABLE I
ACCESS LOGS OF ROSPEEX CLOUD SERVERS FROM DECEMBER 1, 2013 TO MAY 31, 2014.

	Speech recognition	Speech synthesis
Num of requests	10445	23519
Processing time [msec] (median)	961	399
RTF (median)	0.760	0.192

TABLE II
CATEGORIZED UTTERANCES FROM JANUARY 1, 2014 TO NOVEMBER 28, 2014.

Category	# of utterances	Ratio[%]
1. Basic communication	1894	31.70
2. Q & A	1153	19.30
3. Motion control	258	4.31
4. Controling appliances	229	3.83
5. Learning & recognition	215	3.59
6. Other control commands	41	0.68
7. Fragmented speech	2205	36.91
Total	5995	100

V. DISCUSSIONS

A. Reducing the Waiting Time

In Fig. 5, some requests took more than 1000 [msec]. This waiting time is harmful because it might deteriorate the interactive aspect of dialogues. Here we investigate the reason and provide a potential solution.

Currently, the rospeex module starts sending the speech file to the rospeex server when the end of the speech is detected. On the other hand, we can reduce the waiting time if the rospeex module segments the speech into several fragments and sends them fragment by fragment. In this approach, the rospeex module starts sending the Nth fragment when it is obtained. When the end of the speech is detected, the recognition process will be almost finished, and the result is immediately returned to the rospeex module. In future work, we plan to reduce the total waiting time by the above approach. Although network communication time also affects the total waiting time, this is beyond the scope of this paper.

B. Advantages/Disadvantages of Smartphone Interfaces

From the viewpoint of applying speech recognition to robots, smartphone interfaces have several advantages over on-board microphone interfaces.

- Availability
  For most roboticists, finding smartphones is far easier than finding microphones that have desirable speech recognition characteristics.
- Better SNR
  When a speaker talks by a smartphone, the distance between the speaker and microphone is usually shorter than on-board microphone interfaces. This improves the SNR and leads to better performance.
  In the standard setup of the RoboCup@Home competitions, since the distance between the speaker and the robot is usually about 1 [m], the SNR is very small

because the sound is attenuated as the 2nd order of distance.

- Avoiding addressee/side-participant recognition [18]
  When there are multiple robots in the multi-party conversation, they have to recognize which utterance is directed to it [23]. This problem remains unless non-speech features are used. On the other hand, we can avoid this problem by having buttons on the smartphone interface.

Smartphone interfaces for robots have the following disadvantages:

- Non-hands-free interface
  In most cases, speakers tend to expect that the robot can understand what they are talking about without additional devices. Having an additional device might not be accepted.
- No difference with bluetooth microphones
  It is true that bluetooth microphones have advantages over smartphones, such as weight. In our experience, however, bluetooth microphones are stable when connected to smartphones but unstable when connected to PCs.

C. Building Cloud Services for Robots and Smartphones

Suppose a scientist starts to provide a cloud service. After obtaining many active users (developers), even a small amount of maintenance time might affect them. Continuous monitoring is required to avoid this; however, scientists are unlikely to have incentive to dedicate themselves only for monitoring. Thus it is desirable for the service to be profitable enough to hire operators.

Therefore, making the service applicable to both robots and other devices is important. Limiting the market to robots will not become an incentive for many companies. Actually, our cloud engines were first launched for smartphone applications [21]. Many mash-ups exist that use rospeex, such as virtual agents.

VI. Conclusion

In this paper, we presented a cloud platform called rospeex that enables roboticists to build multilingual conversational robots. Instead of merely using cloud services, we built our own cloud services and allow roboticists to use them by robot middleware without payment or authentication. We analyzed the logs which we collected by operating rospeex for over a year. One of our key contributions is that we made it possible to store a huge amount of robotics-related utterances on the cloud server. Future directions include the improvement of language models for robotic tasks, and sharing the log corpus with other roboticists.

VII. Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 24700188.

References

[1] L. Iocchi and T. van der Zant, "RoboCup@Home: Adaptive Benchmarking of Robot Bodies and Minds," in *Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots*, 2010, pp. 171–182.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An Open-Source Robot Operating System," in *ICRA workshop on open source software*, 2009.

[3] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)," in *Proc. IROS*, 2005, pp. 3933–3938.

[4] Y. Matsusaka, H. Asoh, I. Hara, and F. Asano, "Specification and implementation of open source software suite for realizing communication intelligence," *Journal of Robotics and Mechatronics*, vol. 24, no. 1, p. 86, 2012.

[5] K. Nakadai, T. Takahashi, H. G. Okuno, H. Nakajima, Y. Hasegawa, and H. Tsujino, "Design and Implementation of Robot Audition System'HARK'Open Source Software for Listening to Three Simultaneous Speakers," *Advanced Robotics*, vol. 24, no. 5-6, pp. 739–761, 2010.

[6] M. Inaba, "Remote-Brained Robotics: Interfacing AI with Real World Behaviors," in *Proc. of the 6th Int. Symp. of Robotics*, 1993, pp. 335–344.

[7] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud Networked Robotics," *Network, IEEE*, vol. 26, no. 3, pp. 28–34, 2012.

[8] J. Kuffner, "Cloud-Enabled Robots," in *Proc. Humanoids*, 2010.

[9] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 2, 2015.

[10] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "DAvinCi: A Cloud Computing Framework for Service Robots," in *Proc. ICRA*, 2010, pp. 3084–3089.

[11] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-Based Robot Grasping with the Google Object Recognition Engine," Proc. ICRA, 2013.

[12] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth Language: Representing and Exchanging Knowledge about Actions, Objects, and Environments," in *Proc. ICRA*, 2012, pp. 1284–1289.

[13] [Online]. Available: http://julius.sourceforge.jp/

[14] [Online]. Available: http://www.cstr.ed.ac.uk/projects/festival/

[15] [Online]. Available: http://open-jtalk.sp.nitech.ac.jp/

[16] K. Haverlock and S. Sudarsan, "Creating Cognitive Applications Powered by IBM Watson: Getting started with the API," IBM, Tech. Rep., 2013.

[17] K. Tsujino, Y. Nakashima, S. Iizuka, and Y. Isoda, "Speech recognition and spoken language understanding for mobile personal assistants: A case study of "shabette concier"," in *Proc. Workshop on Field Speech and Data Management*, 2013, pp. 225–228.

[18] D. Traum, "Issues in Multiparty Dialogues," in *Advances in Agent Communication*. Springer, 2004, pp. 201–211.

[19] M. Fujimoto and S. Nakamura, "Sequential Non-Stationary Noise Tracking Using Particle Filtering with Switching Dynamical System," in *Proceeding of 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, 2006, pp. 769–772.

[20] T. Nakamura, K. Sugiura, T. Nagai, N. Iwahashi, T. Toda, H. Okada, and T. Omori, "Learning Novel Objects for Extended Mobile Manipulation," *Journal of Intelligent & Robotic Systems*, vol. 66, pp. 187–204, 2012.

[21] S. Matsuda, X. Hu, Y. Shiga, H. Kashioka, C. Hori, K. Yasuda, H. Okuma, M. Uchiyama, E. Sumita, H. Kawai, and S. Nakamura, "Multilingual Speech-to-Speech Translation System "VoiceTra"," in *Proc. Workshop on Field Speech and Mobile Data*, 2013, pp. 229–233.

[22] K. Sugiura, Y. Shiga, H. Kawai, T. Misu, and C. Hori, "Non-Monologue HMM-Based Speech Synthesis for Service Robots: A Cloud Robotics Approach," in *Proc. ICRA*, 2014, pp. 2237–2242.

[23] X. Zuo, N. Iwahashi, T. Ryo, S. Matsuda, K. Sugiura, K. Funakoshi, M. Nakano, and N. Oka, "Detection of Robot-Directed Speech by Situated Understanding in Physical Interaction," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 25, pp. 670–682, 2010.