VSP

*Advanced Robotics 24 (2010) 739–761*

ADVANCED ROBOTICS
brill.nl/ar

*Full paper*

# Design and Implementation of Robot Audition System 'HARK' — Open Source Software for Listening to Three Simultaneous Speakers

**Kazuhiro Nakadai** [a,b,*], **Toru Takahashi** [c], **Hiroshi G. Okuno** [c], **Hirofumi Nakajima** [a], **Yuji Hasegawa** [a] **and Hiroshi Tsujino** [a]

[a] Honda Research Institute Japan Co., Ltd, 8-1 Honcho, Wako-shi, Saitama 351-0114, Japan
[b] Tokyo Insititute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
[c] Kyoto University, Yoshidahonchou, Sakyou-ku, Kyoto 606-8501, Japan

Received 2 June 2009; accepted 4 August 2009

**Abstract**
This paper presents the design and implementation of the HARK robot audition software system consisting of sound source localization modules, sound source separation modules and automatic speech recognition modules of separated speech signals that works on any robot with any microphone configuration. Since a robot with ears may be deployed to various auditory environments, the robot audition system should provide an easy way to adapt to them. HARK provides a set of modules to cope with various auditory environments by using an open-sourced middleware, FlowDesigner, and reduces the overheads of data transfer between modules. HARK has been open-sourced since April 2008. The resulting implementation of HARK with MUSIC-based sound source localization, GSS-based sound source separation and Missing Feature Theory-based automatic speech recognition on Honda ASIMO, SIG2 and Robovie R2 attains recognizing three simultaneous utterances with the delay of 1.9 s at the word correct rate of 80–90% for three speakers.


**Keywords**
Robot audition, open source software, sound source localization, sound source separation, automatic speech recognition

## 1. Introduction

Speech recognition is essential for communication and social interaction, and people with normal hearing capabilities can listen to many kinds of sounds under various acoustic conditions. Robots should have a hearing capability equivalent to

* To whom correspondence should be addressed. E-mail: nakadai@jp.honda-ri.com

 DOI:10.1163/016918610X493561

ours to realize human–robot communication and social interaction, when they are expected to help us in a daily environment. In such a daily environment, a lot of noise sources, including the robot's own motor noises, exist besides target speech sources. Most robot systems for social interaction by speech avoided this problem by forcing the attendants of interaction to wear a headset microphone [1]. For smoother and more natural interactions, a robot should listen to sounds with its own ears instead of using attendants' headset microphones. The concept of 'robot audition', which realizes recognition of noisy speech such as simultaneous speakers by using robot-embedded microphones, i.e., the ears of a robot, was proposed in Ref. [2]. It has been studied actively for recent years, as typified by organized sessions on robot audition at the International Conferences on Intelligent Robots and Systems (IROS 2004–2009), and a special session on robot audition at the IEEE International Conference on Acousitcs Speech and Signal Processing (ICASSP 2009) of the Signal Processing Society.

Some robots have been equipped with hearing capabilities: sound source localization (i.e., identification of the direction where the sound originates), sound source separation (i.e., separation of sound signals of a consistent property) and recognition of separated sounds. Hadaly of Waseda University [3] can localize the speaker as well as recognize speech by automatic speech recognition (ASR). Hadaly uses a microphone array for sound source localization, but the microphone array is mounted in the body and its absolute position is fixed during head movements. Sound source separation is not exploited and a microphone for speech recognition is attached to the speaker. Jijo-2 [4] can recognize a phrase command by a speech-recognition system. Jijo-2 uses its microphone for speech recognition, but it first stops, listens to the speaker and recognizes what he/she says. That is, Jijo-2 lacks sound source localization or sound source separation — it just exploits the speech enhancement.

At international conferences, like IROS and ICASSP as mentioned above, most researchers on robot audition have focused on sound source localization and sound source separation. Asano *et al.* have developed sound source localization and separation by integrating audio and visual information by a particle filter and implemented it on HRP-2 with eight microphones [5]. HRP-2 can recognize one speaker's utterance under noisy or interfering speakers. Nakadai *et al.* have developed binaural sound source localization and separation by using active binaural hearing and implemented it on SIG [6]. SIG can recognize three simultaneous speakers' utterances. Yamamoto *et al.* also have proposed a new speech recognition method based on the Missing Feature Theory (MFT) [7]. It uses a map of reliability in the temporal-frequency domain to show the remarkable improvement of speech recognition of separated sounds. Kim *et al.* have developed another binaural sound source localization and separation by integrating sound source localization obtained by CSP (Cross-power Spectrum Phase) and one obtained by visual information with the EM algorithm [8]. This system assumes that at most one predominant sound exists at each time frame. Valin *et al.* have developed sound source localization and

separation by Geometric Source Separation and a multi-channel post-filter with eight microphones to perform speaker tracking [9, 10]. This system was implemented on Spartacus [11]. Yamamoto *et al.* have developed an automatic missing feature mask (MFM) generation by taking the property of a multi-channel post-filter and environmental noises into consideration [12]. Valin released the sound source localization and separation called 'ManyEars' as GPL open source software. Yamamoto and Nakadai have realized a lot of improvements of their robot audition system by evaluating with various kinds of benchmarks including listening to three simultaneous speakers on various kinds of microphone configurations for three kinds of robots under different acoustic conditions. However, the response time was about 10 s, which is not sufficient for smooth human–robot interactions. Since the robot will be deployed to various acoustic environments with different microphone configurations on different robots, the robot audition software should provide an easy way to adapt to each setting.

The requirements on robot audition software are summarized as:

(i) It should localize, separate and recognize sound sources robustly even when multiple speeches and other noise sources exist simultaneously.

(ii) It provides a set of modules for auditory processing including sound source localization, sound source separation, ASR, sound input devices and other miscellaneous functions.

(iii) It provides an easy way to choose and combine various kinds of modules.

(iv) It supports real-time processing or at least minimizes the time of delay by using a mechanism to share acoustic data between modules.

(v) It should have high usability so that various kinds of researchers and developpers can use it.

This paper presents the design and implementation of our open-sourced robot audition software 'HARK' that satisfies the above requirements. Here, HARK (which means 'listen' in old English) stands for Honda Research Institute Japan Audition for Robots with Kyoto University (http://winnie.kuis.kyoto-u.ac.jp/HARK/).

HARK has the capability to listen to three simultaneous speeches by integrating preprocessing and ASR based on MFT. It provides a complete module set consisting of sound localization, tracking, separation and speech recognition to make a real-time robot audition system without any computer programming skill. Many multi-channel sound cards are supported to build a real-time robot audition system easily. Modules for preprocessing such as sound source localization, tracking and separation that were used in our reported systems are available. The modules for preprocessing can be integrated with ASR based on Missing Feature Theory. Thanks to its module-based architecture, it has general applicability to various types of robots and hardware configurations in real-time. It provides three user levels of robot audition system construction: implementation of each module with C++ for

core developers, network construction using prepared modules for researchers in robot systems, and system customization by changing parameters in provided networks of robot audition systems for beginners and robot system engineers. For the core developers, they can easily migrate their systems to HARK modules, because every HARK module inherits the same super class in C++ and the class has only a few methods that should be implemented by the developers. In addition, once they learn how to implement a module, any other HARK module is able to be implemented in the same way. For the researchers, we will provide a HARK network library, including a lot of sample networks, and all networks will be available on the HARK web site. We found that this kind of library strongly helps the construction of their own robot audition systems from our 1-year educational experience using HARK in universities. For the beginners and the system engineers, the network library with adequate documentation is also effective. In addition, we also provide some sample applications like dialog systems that connect with a HARK-based robot audition system. These applications also show how HARK is easy and flexible in terms of system integration.

In this paper, we describe the design and implementation of HARK, give an overview of a robot audition system with HARK, and evaluate its performance in speech recognition in noisy environments and general applicability with demonstrations. HARK is designed by taking the following two points into consideration:

(i) Evaluation of a technique that each researcher is focusing on in a total robot audition system.

(ii) Construction of a robot system with listening capability for researchers in robotics despite their familiarity with robot audition.

The rest of the paper is organized as follows. Section 2 introduces related work for HARK. Section 3 explains the implementation of HARK. Section 4 shows how to use HARK to construct robot audition systems. Section 5 describes the evaluation of the system. Section 6 concludes the paper.

## 2. Robot Audition Software on Software Programming Environments

In robotics, a variety of software programming environments have been reported [13–16]. Every environment improved software reusability by using a modular architecture. Robot audition software also should be based on a programming environment with a modular architecture and should satisfy the above-mentioned five requirements. However, most programming environments do not provide audio-related modules in advance, although some of them support ultrasonic-based localization. For example, MATLAB is commonly used in the signal processing community. Although it is not for robot audition, it is powerful for prototyping because a large number of audio-related functions are prepared in advance. However, it is not open-sourced and sometimes changes its API without any notice. As this causes problems, we have to modify our programs according to the API changes every

time we find the changes. In addition, its processing speed is insufficient for real-time processing because MATLAB focuses on prototyping. Thus, MATLAB satisfies requirements (ii), (iii) and (v). ManyEars (http://manyears.sourceforge.net/) was the only package to focus on robot audition, as far as the authors know. It includes a set of modules for preprocessing using a microphone array such as steered beamforming-based sound source localization [10], particle filtering-based sound source tracking [10] and geometric source separation-based sound source separation [9]. They showed a robot application by connecting ManyEars with a commercial ASR system. However, the total integration of preprocessing and ASR is not easy because of mismatching problems, i.e., ASR assumes clean and non-distorted input signals, while preprocessing outputs distorted and/or noise-contaminated sounds. Therefore, ManyEars lacks an integration mechanism between preprocessing and ASR, and flexible acoustic feature extraction to improve ASR's robustness for noise and distortion. Although ManyEars fully satisfies requirements (iii) and (iv), it partically satisfies other requirements, i.e., it provides only modules for preprocessing, and no network of a robot audition system is provided for beginners and robot system engineers. Thus, no software environment satisfies the five requirements for robot audition so far.

## 3. Design and Implementation of HARK

HARK was developed and released so that the five requirements can be satisfied. It runs on a data flow-oriented software programming environment called FlowDesigner (http://flowdesigner.sourceforge.net/) [17]. It is open-sourced and integrates modules by using shared objects, i.e., function call-based integration. FlowDesigner allows us to create reusable modules and to connect them together using a standardized mechanism to create a network of modules. The modules are connected dynamically at run time. A network composed of multiple modules can be used in other networks. This mechanism makes the whole system easier to maintain since everything is grouped into smaller functional modules. If a program is created by connecting the modules, the user can execute the program from the GUI interface or from a shell terminal. When two modules have matching interfaces, they are able to be connected regardless of their internal processes. One-to-many and many-to-many connections are also possible. A module is coded in programming language C++ and implemented as an inherited class of the fundamental module. Dynamic linking at run time is realized as a shared object in the case of Linux. Since data communication is done by using a pointer, it is much faster than socket communication. Therefore, FlowDesigner maintains a well-balanced trade-off between independence and processing speed. Thus, FlowDesigner satisfies the above two requirements (i) and (ii). Therefore, we decided to use FlowDesigner as a programming environment for HARK.

HARK provides a complete module set consisting of sound localization, tracking, separation and speech recognition to satisfy the last requirement (iii). These

modules run on FlowDesigner, which works on Linux (Fedora distributions are recommended). The modules are classified into eight categories: multi-channel audio I/O, sound source localization and tracking, sound source separation, acoustic feature extraction for ASR, automatic MFM generation, ASR interface, and data conversion and operation. (MFT)-based ASR is also provided as a patch source code for Julius/Julian [18], which is one of the most popular Japanese open source speech recognition systems (recently, Julius started providing an English acoustic model). Only MFT-ASR is implemented as a non-FlowDesigner module in HARK, but it connects with FlowDesigner by using a module in the ASR interface. Table 1 summarizes the categories and modules provided by HARK (the latest HARK version is 0.1.7, but Table 1 is based on a new version of HARK which will be released in the near future after small modifications). Users are able to make their own robot audition system by combining these modules and also connect their existing system with a HARK-based robot audition system by using a socket interface. However, it is sometimes confusing what modules should be connected and how to connect them with other systems. For the user's convenience, thus, the full English documentation and tutorials of HARK are available with sample networks of robot audition systems to avoid such confusion. In addition, for trial use, a live DVD image of HARK is also available on demand.

Modules in HARK are described in detail from the following sections.

### *3.1. Multi-channel Audio Input*

These modules produce an output of multi-channel audio signals. In this category, there are two modules: AudioStreamFromWave and AudioStreamFromMic. The AudioStreamFromWave module reads audio signals from a file of WAVE format (RIFF waveform audio format). The AudioStreamFromMic module captures audio signals using a sound input device. In the AudioStreamFromMic module, three types of devices are available: ALSA (Advanced Linux Sound Architecture)-based sound cards (http://www.alsa-project.org/), RASP series (http://jeol-st.com/mt/2007/04/rasp2.html (in Japanese)) and TD-BD-8CSUSB (http://www.inrevium.jp/eng/news.html). ALSA is an open source sound I/O driver and it supports a lot of sound cards including multi-channel ones such as RME Hammerfall. RASP series are multi-channel audio signal processing devices with CPUs produced by JEOL System Technology. They support both multi-channel audio signal recording and processing. TD-BD-8CSUSB is one of the smallest eight-channel audio input systems for embedded use, which is produced by Tokyo Electron Device. It captures audio signals *via* a USB interface. SaveRawPCM saves audio data as a raw PCM file.

### *3.2. Sound Source Localization and Tracking*

Seven modules are prepared for sound source localization and tracking. LocalizeMUSIC calculates directions of arrival of sound sources from a multi-channel audio signal input, and outputs the number of sound sources and their directions

**Table 1.**
Modules provided by HARK

| Category name | Module name | Brief explanation |
|---|---|---|
| Multi-channel audio I/O | AudioStreamFromMic | capture sound from microphones |
| | AudioStreamFromWave | capture sound from a wave file |
| | SaveRawPCM | save PCM to a file |
| Sound source localization and tracking | LocalizeMUSIC | localize sound sources with a MUSIC algorithm |
| | ConstantLocalization | use specified directions as localization results |
| | SourceTracker | track sound sources from localization results |
| | DisplayLocalization | display localization results on a two-dimensional viewer (time-azimuth) |
| | SaveSourceLocation | save localization results to a file |
| | LoadSourceLocation | load localization results from a file |
| | SourceIntervalExtender | extend tracking results forward |
| Sound source separation | DSBeamformer | separate sound sources with delay-and-sum beamforming |
| | GSS | separate sound sources with geometric source separation |
| | Postfilter | enhance speech signals |
| | BGNEstimator | estimate background noise |
| Acoustic feature extraction | MelFilterBank | apply Mel filter-bank |
| | MFCCExtraction | extract MFCC features |
| | MSLSExtraction | extract MSLS features |
| | SpectralMeanNormalization | apply spectral mean normalization (SMN) |
| | Delta | calculate delta features |
| | FeatureRemover | remove specific features |
| | PreEmphasis | apply pre-emphasis |
| | SaveFeatures | save features to files |
| Automatic MFM generation | MFMGeneration | estimate MFMs |
| | DeltaMask | calculate MFMs for delta features |
| | DeltaPowerMask | calculate MFMs for delta power feature |
| ASR interface | SpeechRecognitionClient | send acoustic features to an ASR system |
| | SpeechRecognitionSMNClient | send acoustic features to an ASR system after SMN |
| MFT-ASR | Multiband Julius/Julian | an ASR system (non-FlowDesigner module) |
| Data conversion and operation | MultiFFT | do frequency analysis for multi-channel data with fast Fourier transform |
| | Synthesize | synthesize a wave signal from a sequence of spectra |
| | WhiteNoiseAdder | add white noise |
| | ChannelSelector | select the specified channels of an input signal |
| | SourceSelectorByDirection | select sound sources within the specified angle range |
| | SourceSelectorByID | select sound sources with the specified IDs |
| | MatrixToMap | convert a matrix structure to a map structure |
| | PowerCalcForMap | calculate power from map-structured data |
| | PowerCalcForMatrix | calculate power from matrix-structured data |

by each time frame. Since this module uses an adaptive beamforming algorithm called MUltiple Signal Classification (MUSIC) [19], it localizes multiple sound sources robustly in real environments. Basically, the MUSIC algorithm requires transfer functions between a sound source and each microphone, which is called steering vectors. To take the effect of a robot's head into account, a steering vector is obtained from a measured impulse response. The impulse responses are repeatedly measured by changing the directions of sound sources like the measurement of head-related transfer functions in binaural processing. This is time-consuming. Thus, HARK provides another method to obtain steering vectors. This method estimates a transfer function from the geometrical relationship between a sound source and each microphone by assuming a free acoustic space. This is provided by 'tftool', which is a non-module tool included in HARK. Since the free acoustic space is assumed in tftool, performance of localization drops a little, but the steering vectors are easily obtained. ConstantLocalization outputs fixed sound directions without localization, and it is mainly used for performance evaluation and debugging. These modules are replaceable with each other. SourceTracker tracks sound sources from the localization results and it outputs sound source directions with source IDs. When the current direction and the previous direction derive from the same sound source, they are given the same source ID. DisplayLocalization is a viewer module for LocalizeMUSIC, ConstantLocalization and SourceTracker. SaveSourceLocation stores localization results to a text file in every time frame. LoadSourceLocation loads source location information from the text file. SourceIntervalExtender extends tracking results forward to deal with start point misdetection of a sound source.

### *3.3. Sound Source Separation*

Three modules are prepared for sound source separation, i.e., GSS, Postfilter and BGNEstimator. GSS is a more sophisticated sound source separation module using a Geometric Source Separation algorithm. GSS is a kind of hybrid algorithm of Blind Source Separation (BSS) [20] and beamforming. It relaxes BSS's limitations such as permutation and scaling problems by introducing 'geometric constraints' obtained from the locations of microphones and sound sources. Our implementation has two unique characteristics for robot audition. One is that the robot's own noises are taken into account. To deal with robot noises such as fans, we can specify a fixed noise direction in GSS. When this is specified, this module always removes the corresponding sound source as a robot's noise in spite of sound source localization results. The other is that sound separation of moving speakers is taken into account. A separation matrix in GSS should be initialized when the location of a speaker is changed, because the separation matrix is generated based on a geometrical relationship between a speaker and each microphone. However, such initialization sometimes causes slow convergence of the separation matrix. Thus, criteria and timing of separation matrix initialization are controllable in our implementation. Currently, we are trying to add two new features to GSS, i.e., adaptive step-size

control that provides faster convergence of the separation matrix [21] and Optima Controlled Recursive Average [22] that controls window size adaptively for better separation. We are testing these features and have some promising results [23]. They will be included in a future HARK release. We also prepared another module for sound source separation called DSBeamformer, which uses a simple delay-and-sum beamformer. The same processing is obtained by setting the stepsize parameter in GSS to 0.

Postfilter is used to enhance the output of GSS [24]. It is a spectral filter using an optimal noise estimator described in Ref. [25]. We extended the original noise estimator to estimate both stationary and non-stationary noise by using multi-channel information, while most post-filters address the reduction of a type of noise — stationary background noise [26]. Another advantage of Postfilter is parameter tuning. Since Postfilter consists of several signal processing techniques, there are a large number of parameters. Such parameters are mutually dependent, and the best parameter setting differs according to the surrounding acoustic environment and situation. Thus, most parameters are able to be controlled from the FlowDesinger GUI/CUI. The best parameter setting is, then, obtained by using a parameter search system based on a genetic algorithm.

BGNEstimator estimates the averaged background noise spectrum that is used for a noise reduction technique called Minima Controlled Recursive Average (MCRA) included in Postfilter.

ManyEars provides SeparGSS combining GSS and a multi-channel post-filter. In this module, the number of controllable parameters is small. Thus, we provide a patch for SeparGSS that changes I/O IF to be able to use as a HARK module. However, a combination of GSS and Postfilter provides a more flexible solution and better performance when we cope with actual situations such as the robot's own noises, simultaneous speech recognition and moving speakers.

### *3.4. Acoustic Feature Extraction for ASR*

Acoustic feature extraction is an advantage of HARK because it is quite flexible to find the best acoustic feature. Everything can be done in a GUI environment although famous packages having feature extraction functions such as HTK (http://htk.eng.cam.ac.uk/) require text file editing. This category includes eight modules: MelFilterBank, MFCCExtraction, MSLSExtraction, Delta, FeatureRemover, SpectralMeanNormalization, PreEmphasis and SaveFeatures. Most conventional ASR systems use a Mel-frequency cepstral coefficient (MFCC) as an acoustic feature. MelFilterBank consumes an input of a spectrum, analyzes it using Mel-frequency filter banks and produces an output of a Mel-scale spectrum. MFCCExtraction extracts the MFCC acoustic feature from the Mel-scale spectrum. However, in the case of MFCC, noises and distortions that are concentrated in some areas in the spectro-temporal space are spread to all coefficients. Thus, HARK provides Mel-scale log spectrum (MSLS) [27] as an alternative acoustic feature. MSLSExtraction consumes an input of the Mel-filter spectrum, performs liftering

and produces an output of MSLS features. SpectralMeanNormalization searches the averaged spectrum with the same direction as the current utterance and then it is subtracted from the current utterance. The averaged spectrum is continuously updated by using the previously detected utterances for each direction at 10° intervals. It works well in real-time ASR systems to improve noise robustness of MSLS features. Finally, Delta consumes an output of the MSLS/MFCC features, calculates the linear regression of each spectral coefficient and produces an output of the MSLS/MFCC features with delta features. FeatureRemover removes any coefficient (e.g., power) in an acoustic feature. PreEmphasis provides a low-pass filter to emphasize speech characteristics in the time or frequency domain according to the user's preference. SaveFeatures stores a sequence of MSLS/MFCC features into a file per utterance.

### *3.5. Automatic MFM Generation*

This category is one of the unique chracteristics in HARK. MFT is an approach for noise-robust ASR [28, 29]. MFT uses only reliable acoustic features in speech recognition, and masks out unreliable parts caused by interfering sounds and preprocessing. MFT, thus, provides smooth integration between preprocessing and ASR. The mask used in MFT is called a MFM represented as a spectro-temporal map. Three modules are prepared to generate MFMs without using prior information.

The inputs of MFMGeneration are Mel-filtered spectra of the separated sound the post-filtered sound and the background noise estimated by the BGNEstimator. It estimates a leak noise by using the fact that the post-filtered sound is similar to clean speech, while the separated sound includes a background and a leak noise. When the estimated leak noise is lower than a specified threshold, such a frequency bin is regarded as reliable, otherwise it is unreliable. MFMGeneration thus produces a MFM as a binary mask. DeltaMask consumes an input of MFMs and produces an output of delta MFMs. DeltaPowerMask generates a MFM for a delta power coefficient if an acoustic feature includes the coefficient.

### *3.6. ASR Interface*

SpeechRecognitionClient consumes inputs of acoustic features, MFMs and sound source directions. It sends them to Multiband Julius/Julian *via* TCP/IP communication. SpeechRecognitionSMNClient has almost the same function as SpeechRecognitionClient. The difference is that it supports SMN. SpeechRecognitionSMNClient uses a current utterance to estimate the averaged spectrum, while SpectralMeanNormalization uses the previously detected utterances. SpeechRecognitionSMNClient estimates more precise spectral mean, but it has to wait until the current utterance finishes. Thus, a robot audition system with SpeechRecognitionSMNClient should be used in offline systems.

### 3.7. MFT-ASR

Multiband Julius/Julian is provided as a patch source code to original Julius/Julian. It supports various types of hidden Markov model (HMMs) such as shared-state triphones and tied-mixture models. Both statistical language model and network grammar are supported. In decoding, an ordered word bi-gram is used in the first pass and a reverse ordered word tri-gram in the second pass. It works as a standalone or client–server application. To run as a server, we modified the system to be able to communicate acoustic features and MFMs *via* a network.

### 3.8. Data Conversion and Operation

We also prepared nine conversion and operation modules. For frequency analysis and re-synthesis, MultiFFT and Synthesize are available. WhiteNoiseAdder adds white noise basically in order to relax distortion problems caused by the Postfilter. ChannelSelector selects any number of channels obtained by AudioStreamFromMic and AudioStreamFromWave. SourceSelectorByDirection selects sound sources within the specified angle range to avoid ASR from recognition of undesirable sound sources. SourceSelectionByID selects separated sound spectra that have the specified speaker ID. MatrixToMap converts matrix data into map data, i.e., matrix data associated with sound source ID. PowerCalcForMap and PowerCalcForMatrix calculate power in each element of a complex float map and matrix, respectively.

## 4. Building a Robot Audition System using HARK

In this section, we describe how to build a robot audition system using HARK. First download HARK files from the website (http://winnie.kuis.kyoto-u.ac.jp/HARK/) and install them. Two types of packages are available from the HARK website — one is tar ball and the other is rpm files for Fedora 7. HARK developers should use the former so that they can specify HARK's install directory by themselves. The latter is convenient for HARK users, although the install directory is fixed.

To build a robot audition system, we need to create networks of modules by connecting HARK modules. A robot audition system that can localize, separate and recognize simultaneous speeches is shown in Fig. 1. This network consists of six sections according to categories in Table 1. The multi-channel audio input section captures sounds from a microphone array (AudioStreamFromMic). The localization and tracking section performs sound source localization and tracking using LocalizeMUSIC, SourceTracker, etc. The separation section performs sound source separation using GSS, Postfilter and some other modules. The acoustic feature extraction section extracts MSLS-based acoustic features. This section is important to improve ASR performance; thus, we divided acoustic feature extraction function into fundamental modules so that users can try various kinds of feature extraction methods. The MFM generation section generates a MFM corresponding to the extracted acoustic feature by using MFMGeneration and DeltaMask. The ASR

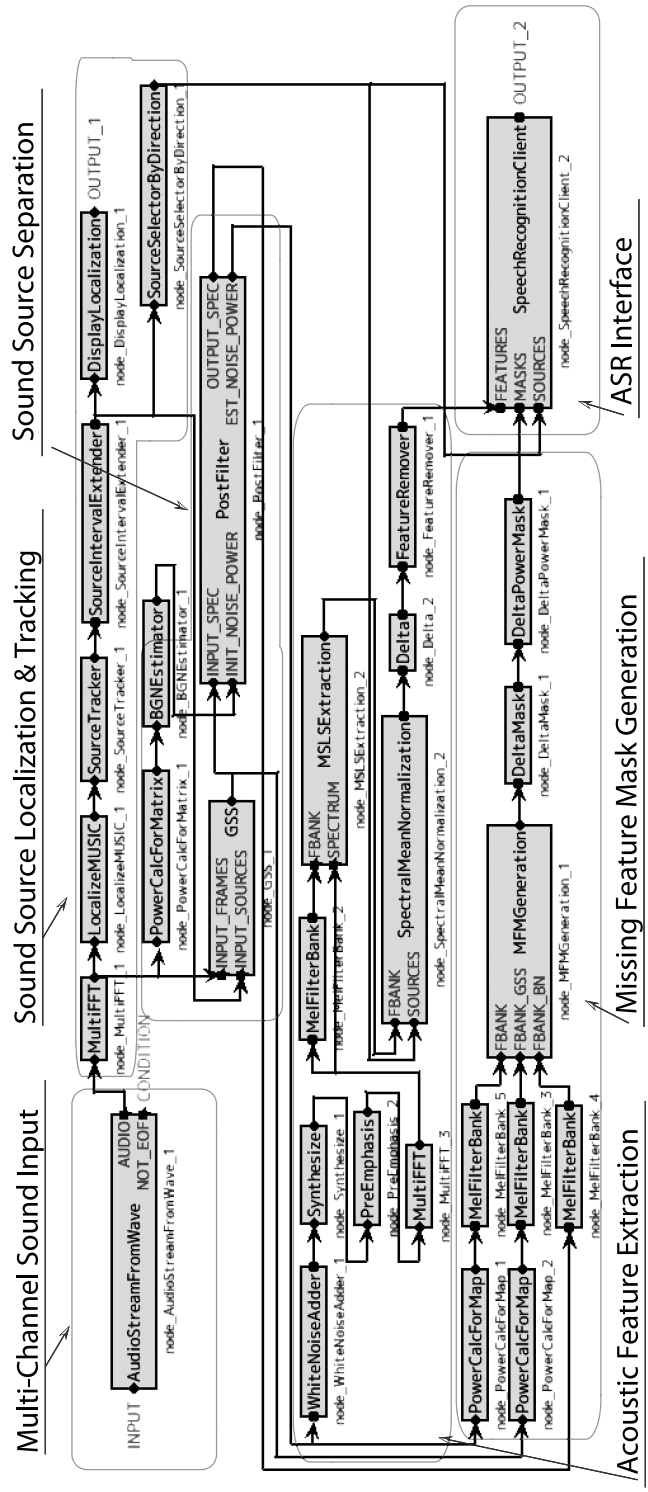



**Figure 1.** Example of a robot audition system using HARK.

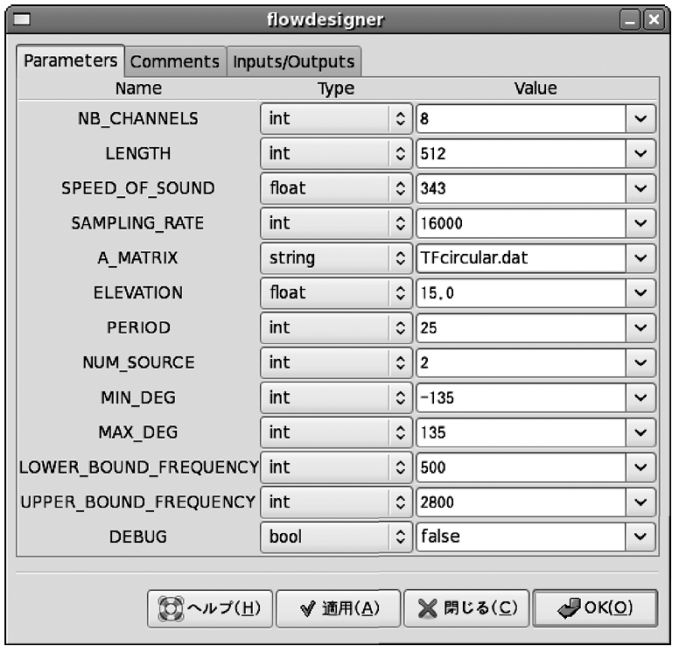



**Figure 2.** Window for property setting in LocalizeMUSIC.

interface section sends a pair of acoustic feature and missing feature mask to MFT-ASR *via* a socket interface (SpeechRecognitionClient). A large variety of parameter settings are possible by using property window in each module as shown in Fig. 2. Some modules are replaceable. When AudioStreamFromMic is replaced with AudioStreamFromWave, input sound source is changed from a microphone array to a wave file. We can flexibly build various robot audition systems by replacing a module with another or add a new module to the network.

## 5. Evaluation

Figure 1 shows a robot audition system using HARK, which we used for evaluation. The system uses MUSIC for sound source localization (LocalizeMUSIC), GSS (GSS), multi-channel post-filter (Postfilter) and speech recognition client for MFT (SpeechRecognitionClient). As shown in Fig. 3, we used four types of hardware configurations — Honda ASIMO (two microphone layouts), SIG2 and Robovie R2. The robot audition system is applied to all configurations by modifying only the specification file of microphone positions. We, then, evaluated HARK in terms of the following points:

(i) Speech recognition performance (ASIMO, SIG2).

(ii) Processing speed (ASIMO).

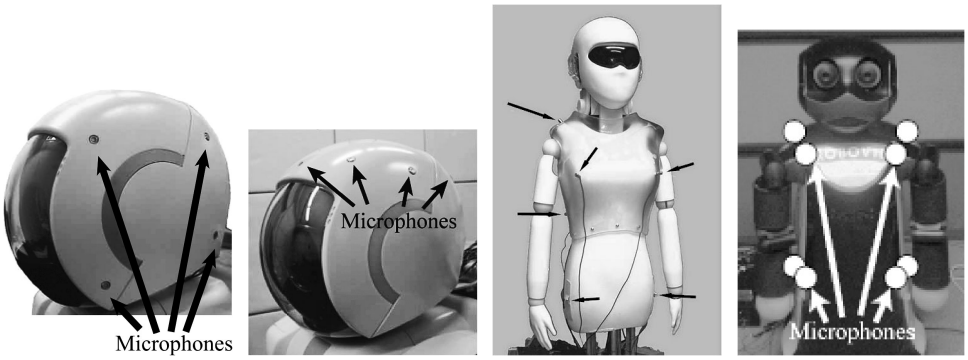



**Figure 3.** Eight-channel microphone arrays. (a) Cube layout (ASIMO). (b) Circular layout (ASIMO). (c) Sparse layout (SIG2). (d) Sparse layout (Robovie).

(iii) General applicability and flexibility of HARK through applications of HARK (ASIMO, Robovie).

### *5.1. Speech Recognition Performance*

Speech recognition performance has been evaluated in terms of robustness for noise, and in terms of effectiveness of preprocessing and MFT-ASR.

#### *5.1.1. Robustness for Noise*

The experimental setting of isolated word recognition has been evaluated using Honda ASIMO with an eight-channel circular microphone array shown in Fig. 3a. ASIMO was located at the center of a room which was 7 m $\times$ 4 m. Three walls were covered with sound absorbing materials, while the other wall was made of glass, which produces strong echoes. The reverberation time (RT20) of the room was about 0.2 s. However, the reverberation was not uniform in the room because of an asymmetrical echo generated by the glass wall. As a test dataset, an ATR phonemically balanced word-set (ATR-PB) was used. This word-set includes 216 Japanese words per person. We used six persons' data from the word-set and the average word correct rate (WCR) was measured. A triphone acoustic model was trained with another speech dataset called JNAS (Japanese Newspaper Article Sentences) corpus which includes 60 h of speech data spoken by 306 male and female speakers. Thus, this speech recognition was a word-open test. Three types of noise environments were selected in this experiment:

(i) ASIMO noise including sounds produced by fans and motors was used. Since this noise is mainly produced inside ASIMO, it is near-field stationary noise and it is also regarded as a kind of directional noise, but it has characteristics of diffuse noise to some extent due to reverberation, resonance and diffraction caused by the robot's body and head.

(ii) Music noise was used as a directional noise. In this case, the ASIMO noise was also added. Thus, this noise environment is regarded as a mixture of two noise sources with robot's noise.

(iii) Speech noise, located 60° left from the front direction of ASIMO, was used instead of the music noise that was used in the second condition. The distance between ASIMO and the speech source was 1 m. This means that this is the case of simultaneous speech recognition with robot's noise.

In every noise environment, the target speech source was set to the front direction and at 1 m away from ASIMO. The level of the target speech source was changed from −8 to 3 dB compared with the ASIMO noise. Speech data for isolated word recognition was synthesized by using impulse responses and noise data that were pre-measured with the microphone array in these noise environments. This means that speech files were the input to the robot audition system. The robot audition system for this kind of simulated input was constructed by replacing AudioStreamFromMic with AudioStreamFromWave in Fig. 1. ASIMO noise is first reduced by specifying a fixed noise direction, i.e., a new feature in GSS, and then Postfilter suppresses the remaining noise with the characteristics of diffuse noise. Music noise and speech noise are mainly reduced with GSS because they are non-near-field and directional noise sources. There are two differences between them. One is that speech noise is more likely to overlap a target speech source because the characteristics of noise and the target are similar. The other is that music is always played while speech alternately continues stopping and playing. This affects the convergence of a separation matrix for GSS and thus flexible update control of the separation matrix is necessary. Therefore, speech noise is a more difficult noise than music noise.

Figure 4 shows the results for three kinds of noise conditions. To determine the baseline performance, we measured WCR by selecting the front microphone without any preprocessing. Figure 4a shows that the robot audition system drastically improves the performance. Even when SNR is as low as −6 dB, WCR still maintains 90%. This indicates that the post-filtering process in the sound source separation module deals with the characteristics of diffuse noise properly. From Fig. 4b and 4c, WCRs without any preprocessing were almost 0%. As target speech source is contaminated by both ASIMO noise and another directional noise, it is difficult to recognize such noisy speech without preprocessing. A HARK-based robot audition system improved speech recognition drastically. The case of speech noise is slightly worse than that of music noise, but in both cases, the WCRs were over 75% for signal-to-noise ratios higher than −4 dB in both cases. GSS worked for directional noise sources regardless of noise types.

*5.1.2. Effectiveness of Preprocessing and MFT-ASR*

To determine the effectiveness of each technique in the robot audition system, we conducted isolated word recognition of three simultaneous speeches. SIG2 with an eight-channel microphone array shown in Fig. 3c was used in a 4 m × 5 m room

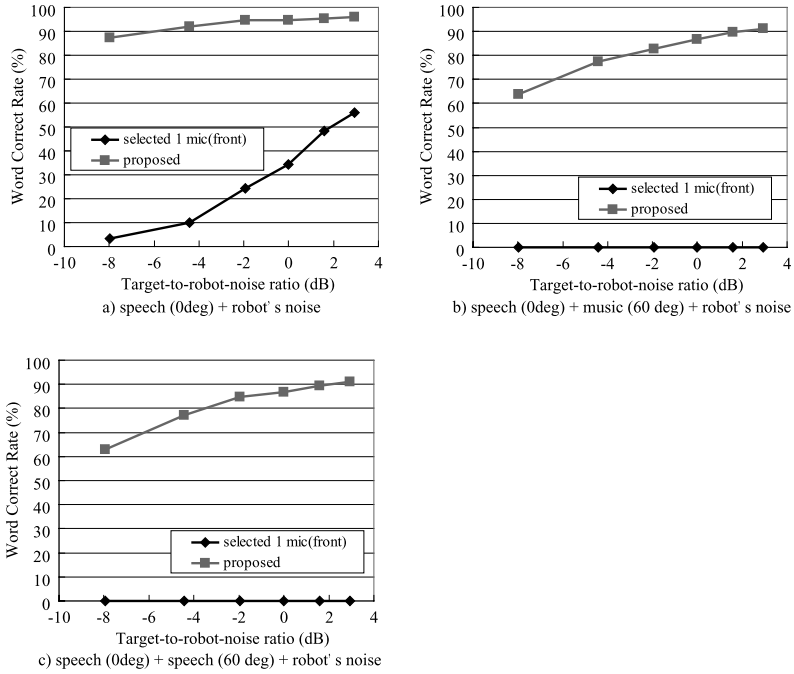



**Figure 4.** Isolated word recognition (WCR) against diffuse and directional noise.

with 0.3–0.4 s of RT20. Three simultaneous speeches for test data were recorded with the eight-channed microphone array in the room by using three loudspeakers (Genelec 1029A). The distance between each loudspeaker and the center of the robot was 2 m. Two types of speaker settings, i.e., (−30°, 0°, 30°) and (−60°, 0°, 60°) were evaluated (0° means the front direction of SIG2). As a test dataset, a female (f101), a male (m101) and another male (m102) speech source were selected from ATR-PB for the left, center and right loudspeakers, respectively. Three words for simultaneous speech were selected at random. In this recording, the power for the robot was turned off. By using the test data, the system recognized the three speakers with the following five conditions:

C1 The input from the left-front microphone was used without any preprocessing using a clean acoustic model.

C2 GSS and post-filter were used as preprocessing, but the MFT function was not. The clean acoustic model was used.

C3 The same condition as C2 was used except for the use of the MFT function with automatically generated MFMs.

C4 The multi-condition-trained acoustic model was used. The other conditions were the same as C3.

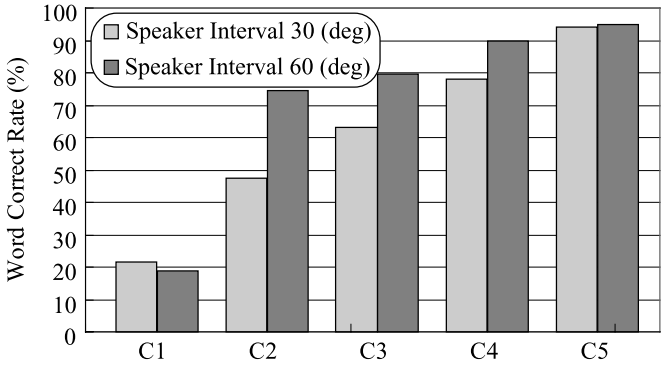



**Figure 5.** Effectiveness of preprocessing and MFT-ASR.

C5 The same condition as C4 was used except for the use of *a priori* MFMs created by clean speech. This is regarded as the potential upper limit of our system.

The clean tri-phone-based acoustic model was trained with 10 male and 12 female ATR-PB word-sets excluding the three word-sets (f101, m101 and m102) that were used for the recording. Thus, it was a speaker-open and word-closed acoustic model. The multi-condition-trained acoustic model was also a triphone-based HMM, and it was trained with the same ATR-PB word-sets as mentioned above and separated speech datasets. Sound source separation is generally an ill-posed problem and separated sound includes inevitable noises to some extent. To deal with such a noise, noise adaptation techniques such as multi-condition training is effective. This is why we used the multi-condition-trained acoustic model in C4 and C5. The latter sets were generated by separating three-word combinations of f102–m103–m104 and f102–m105–m106, which were recorded in the same way as the test data.

Figure 5 shows WCRs for the front speaker. This showed that every technique contributed to speech recognition performance. WCR was around 90% in the 60° interval case and sound source separation was the most effective, because GSS-based sound source separation works very well. On the other hand, 30-point degradation was found with C2 in the 30° interval case, because sound sources were close to each other. However, the MFT-ASR and multi-condition-trained acoustic model compensated this degradation and improved WCR upto around 80%. The fact that an *a priori* mask showed quite a high performance may suggest many possibilities to improve the algorithms of MFM generation. Note that localization does not affect WCR so much, because we found that localization degrades only 5 points in WCR compared with manually given localization from another experiment.

### *5.2. Processing Time*

As for processing speed, we measured processing time when our robot audition system separated and recognized speech signals of 800 s consisting of isolated words. It took 499 s for our robot audition system to recognize the speech signal with a Pentium 4/2.4 GHz processor. FlowDesigner consumed 369 s and Multiband

**Table 2.**
CPU occupancy of each process

| | Localization | Separation | Feature extraction | Miscellanesus |
|---|---|---|---|---|
| Occupancy (%) | 44.0 | 47.3 | 7.8 | 0.9 |

Julius consumed the remaining time. An occupancy rate of each process running on FlowDesigner is shown in Table 2. Sound source localization and separation consumed most of the computational power. This is because eigenvalue decomposition for localization and update of the decomposition matrix for separation requires a lot of multiplications. This could be reduced by using special hardware like a DSP or FPGA. The delay between the beginning of localization and the beginning of recognition was about 0.40 s, and the delay between the end of separation and the end of recognition was about 0.046 s. On the whole, our robot audition system worked in real-time.

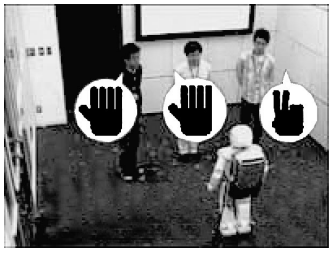
### *5.3. Applications of HARK*

We introduce two applications by using HARK. One is a robot referee for a Rock–Paper–Scissors (RPS) sound game. The other is a meal order taking robot. In both applications, HARK-based robot audition systems were implemented on a laptop PC with a Core 2 Duo/2 GHz processor. The first one is implemented by using ASIMO with another microphone array configuration shown in Fig. 3b. To implement the robot referee, the robot audition system was connected with a simple speech dialog system. It is implemented as a system-initiative dialog system for multiple players based on deterministic finite automata. It first waits for a trigger command to start an RPS sound game, controls the dialog with players in the game and finally decides the winner of the game. Figure 6 shows the snapshots of this referee task. For the moment, this system supports up to three players. In the case with two players, we attained a 70% task completion rate for the games on average [30]. It takes around 2 s to estimate the number of simultaneous players, because the system needs time to detect that all players have finished their RPS utterances. Thus, the system latency was 2–3 s for this task.

The second application also deals with simultaneous speech to take meal orders. In this case, we used Robovie with a microphone array shown in Fig. 3d. In this case, a robot audition system was connected with a simple dialog system for meal order taking. Figure 7 shows snapshots of the meal-order-taking task. In this task, latency from the end of a user's utterance to the beginning of a robot's response was only 1.9 s. This latency was mainly caused by speech recognition because ASR outputs recognition results after the system detects the end of speech. We also constructed the same application using ManyEars. However, the interface between ManyEars and ASR was not socket-based but file-based. Thus, the latency of this task was 7.9 s. The response was 4 times faster using HARK.


a) U2: Let's play an RPS game.


b) U1-3: rock-paper-scissors...



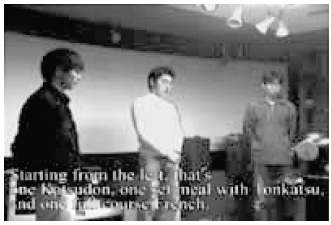
c) U1:paper, U2:paper, U3:scissors


d) ASIMO: U3 won.

**Figure 6.** Snapshots of RPS (A: ASIMO; U1: left user; U2: center user; U3: right user).


a) R: May I take your order.

b) U1-3: order dishes simultaneously.



c) R: repeat the order for confirmation.


d) R: tell the total price.

**Figure 7.** Snapshots of meal-order-taking task (R: Robovie; U1: left user; U2: center user; U3: right user).

## 6. Conclusions

We described an open source robot audition system called HARK, which realizes performance and real-time robot audition systems with a small amount of prior information. HARK provides a complete module set to construct a robot audition

system including a multi-channel input module supporting many sound input devices. It uses modular architecture based on an open source software programming environment, FlowDesigner, which provides module integration with a small latency for real-time processing. Thus, HARK satisfied three requirements for robot audition software, i.e., open source, real-time processing, and many prepared audio modules. Therefore, we can say that HARK is suitable for robot audition. In addition, HARK takes general applicability into account. Thus, HARK is applied to various robots and hardware configurations such as a microphone layout with minimum modification.

These advantages were confirmed through performance evaluation of speech recognition in noisy environments, processing speed and two applications of HARK by combining them with dialog systems by using Honda ASIMO, SIG2 and Robovie.

We believe that HARK is helpful for rapid prototyping of robot audition systems, and also contributes to dialog, vision and navigation research in terms of introducing high-performace robot audition to their system. The extensions of HARK such as a sound scene viewer, sound source identification and music recognition are the subject of future works.

*Acknowledgements*

We thank Drs Jean-Marc Valin and Shunichi Yamamoto for their valuable advice on HARK implementation.

## References

1. C. Breazeal, Emotive qualities in robot speech, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Maui, HI, pp. 1389–1394 (2001).
2. K. Nakadai, T. Lourens, H. G. Okuno and H. Kitano, Active audition for humanoid, in: *Proc. 17th Natl. Conf. on Artificial Intelligence*, Austin, TX, pp. 832–839 (2000).
3. S. Hashimoto, S. Narita, H. Kasahara, A. Takanishi, S. Sugano, K. Shirai, T. Kobayashi, H. Takanobu, T. Kurata, K. Fujiwara, T. Matsuno, T. Kawasaki and K. Hoashi, Humanoid robot — development of an information assistant robot Hadaly, in: *Proc. 6th IEEE Int. Workshop on Robot and Human Communication*, Sendai, pp. 106–111 (1997).
4. H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho and T. Matsui, Socially embedded learning of the office-conversant mobile robot *jijo-2*, in: *Proc. 15th Int. Joint Conf. on Artificial Intelligence*, Nagaya, vol. 1, pp. 880–885 (1997).
5. I. Hara, F. Asano, H. Asoh, J. Ogata, N. Ichimura, Y. Kawai, F. Kanehiro, H. Hirukawa and K. Yamamoto, Robust speech interface based on audio and video information fusion for humanoid HRP-2, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, pp. 2404–2410 (2004).
6. K. Nakadai, D. Matsuura, H. G. Okuno and H. Tsujino, Improvement of recognition of simultaneous speech signals using av integration and scattering theory for humanoid robots, *Speech Commun.* **44**, 97–112 (2004).

7. S. Yamamoto, J.-M. Valin, K. Nakadai, T. Ogata and H. G. Okuno, Enhanced robot speech recognition based on microphone array source separation and missing feature theory, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, pp. 1489–1494 (2005).
8. H.-D. Kim, K. Komatani, T. Ogata and H. G. Okuno, Human tracking system integrating sound and face localization using em algorithm in real environments, *Adv. Robotics* **23**, 629–653 (2007).
9. J.-M. Valin, J. Rouat and F. Michaud, Enhanced robot audition based on microphone array source separation with post-filter, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, pp. 2133–2128 (2004).
10. J.-M. Valin, F. Michaud and J. Rouat, Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering, *Robotics Autonomous Syst. J.* **55**, 216–228 (2007).
11. F. Michaud, C. Côté, D. Létourneau, J.-M. Valin, E. Beaudry, C. Räievsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguére, M.-A. Roux, S. Caron, P. Frenette and F. Kabanza, Robust recognition of simultaneous speech by a mobile robot, *IEEE Trans. Robotics* **23**, 742–752 (2007).
12. S. Yamamoto, K. Nakadai, M. Nakano, H. Tsujino, J.-M. Valin, K. Komatani, T. Ogata and H. G. Okuno, Real-time robot audition system that recognizes simultaneous speech in the real world, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, pp. 5333–5338 (2006).
13. M. Montemerlo, N. Roy and S. Thrun, Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) toolkit, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, pp. 2436–2441 (2006).
14. N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W.-K. Yoon, RT-middleware: distributed component middleware for RT (robot technology), in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, pp. 3555–3560 (2005).
15. C. Schlegel, A component approach for robotics software: communication patterns in the orocos context, *Autonome Mobile Systeme* **18**, 253–263 (2003).
16. C. Côté, Y. Brosseau, D. Létourneau, C. Räievsky and F. Michaud, Using MARIE in software development and integration for autonomous mobile robotics, *Int. J. Adv. Robotic Syst. (Special Issue on Software Development and Integration in Robotics)* **3**, 55–60 (2006).
17. C. Côté, D. Létourneau, F. Michaud, J.-M. Valin, Y. Brosseau, C. Räievsky, M. Lemay and V. Tran, Reusability tools for programming mobile robots, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, pp. 1820–1825 (2004).
18. T. Kawahara and A. Lee, Free software toolkit for Japanese large vocabulary continuous speech recognition, in: *Proc. Int. Conf. on Spoken Language Processing*, Beijing, vol. 4, pp. 476–479 (2000).
19. F. Asano, H. Asoh and T. Matsui, Sound source localization and signal separation for office robot 'Jijo-2', in: *Proc. IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, Taipei, pp. 243–248 (1999).
20. L. C. Parra and C. V. Alvino, Geometric source separation: merging convolutive source separation with geometric beamforming, *IEEE Trans. Speech Audio Process.* **10**, 352–362 (2002).
21. H. Nakajima, K. Nakadai, Y. Hasegawa and H. Tsujino, Adaptive step-size parameter control for real-world blind source separation, in: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Las Vegas, NV, pp. 149–152 (2008).
22. H. Nakajima, K. Nakadai, Y. Hasegawa and H. Tsujino, High performance sound source separation adaptable to environmental changes for robot audition, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, pp. 2165–2171 (2008).

23. K. Nakadai, H. Nakajima, Y. Hasegawa and H. Tsujino, Sound source separation of moving speakers for robot audition, in: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Taipei, pp. 3685–3688 (2009).
24. S. Yamamoto, K. Nakadai, J.-M. Valin, J. Rouat, F. Michaud, K. Komatani, T. Ogata and H. G. Okuno, Making a robot recognize three simultaneous sentences in real-time, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, pp. 897–902 (2005).
25. Y. Ephraim and D. Malah, Speech enhancement using minimum mean-square error short-time spectral amplitude estimator, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32**, 1109–1121 (1984).
26. I. McCowan and H. Bourlard, Microphone array post-filter for diffuse noise field, in: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Orlando, FL, vol. 1, pp. 905–908 (2002).
27. Y. Nishimura, T. Shinozaki, K. Iwano and S. Furui, Noise-robust speech recognition using multiband spectral features, in: *Proc. 148th Acoustical Soc. of America Meet.*, San Diego, CA, no. 1aSC7 (2004).
28. J. Barker, M. Cooke and P. Green, Robust ASR based on clean speech models: an evaluation of missing data techniques for connected digit recognition in noise, in: *Proc. Eurospeech*, Aalborg, pp. 213–216 (2001).
29. M. Cooke, P. Green, L. Josifovski and A. Vizinho, Robust automatic speech recognition with missing and unreliable acoustic data, *Speech Commun.* **34**, 267–285 (2000).
30. K. Nakadai, S. Yamamoto, H. G. Okuno, H. Nakajima, Y. Hasegawa and H. Tsujino, A robot referee for rock–paper–scissors sound games, in: *Proc. IEEE–RAS Int. Conf. on Robots and Automation*, Pasadena, CA, pp. 3469–3474 (2008).

## About the Authors

**Kazuhiro Nakadai** received the BE degree in Electrical Engineering, in 1993, the ME degree in Information Engineering, in 1995, and the PhD degree in Electrical Engineering, in 2003, from The University of Tokyo. He worked with Nippon Telegraph and Telephone and NTT Comware Corp. as a System Engineer, from 1995 to 1999. He was a Researher at Kitano Symbiotic Systems Project, ERATO, Japan Science and Technology Agency (JST), from 1999 to 2003. He is currently a Senior Researcher for Honda Research Institute Japan, Co., Ltd. Since 2006, he is also Visiting Associate Professor at Tokyo Institute of Technology. His research interests include AI, robotics, signal processing, computational auditory scene analysis, multi-modal integration and robot audition. He is a Member of the RSJ, JSAI, ASJ and IEEE.

**Toru Takahashi** received the BE degree in Computer Science, and the ME and DE degrees in Electrical and Electronic Engineering from the Nagoya Institute of Technology, Nagoya, Japan, in 1996, 1998 and 2004, respectively. He had worked as a Research Assistant for about 4 years in Wakayama University. He has been working for Kyoto University since February 2008. His research interests include human–robot interaction, signal processing and speech communication. He is a Member of the IEEE, IEICE, IPSJ, ASJ and RSJ.

**Hiroshi G. Okuno** received the BA and PhD degrees from the University of Tokyo, Japan, in 1972 and 1996, respectively. He worked for Nippon Telegraph and Telephone, JST Kitano Symbiotic Systems Project, and Tokyo University of Science. He is currently a Professor in the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Kyoto, Japan. He was a Visiting Scholar at Stanford University, Stanford, CA, and Visiting Associate Professor at the University of Tokyo. He has done research in programming languages, parallel processing and reasoning mechanisms in AI, and is currently engaged in computational auditory scene analysis, music scene analysis and robot audition. He edited (with D. Rosenthal) *Computational Auditory Scene Analysis* (Lawrence Erlbaum, 1998) and (with T. Yuasa) *Advanced Lisp Technology* (Taylor & Francis, 2002). He has received various awards including the 1990 Best Paper Award of JSAI, the Best Paper Award of IEA/AIE-2001 and 2005, and IEEE/RSJ Nakamura Award for IROS-2001, and 2006 Best Paper Nomination Finalist and IROS2008 Award for Entertainment Robots and Systems Nomination Finalist. He is a Member of the IPSJ, JSAI, JSSST, JSCS, RSJ, ACM, IEEE, AAAI, ASA and ISCA.

**Hirofumi Nakajima** received the BE, ME and PhD degrees from Kogakuin University, in 1994, 1996 and 2007, respectively. He worked with Nittobo Acoustic Engineering Co., Ltd, from 1996 to 2006. He is currently Senior Researcher for Honda Research Institute Japan, Co., Ltd. His research focuses on acoustical signal processing. He is a Member of the ASJ, ASA, JSAI, RSJ and IEEE.

**Yuji Hasegawa** received a BS degree from Tokyo Metropolitan University. He is a Principal Engineer in Honda Research Institute Japan Co., Ltd. His research focuses on bioinspired computation, insect vision, space perception, robot navigation and computer engineering applications.

**Hiroshi Tsujino** is a Chief Researcher at the Honda Research Institute Japan, where he directs the associative interacting intelligence involving brain-like computing, brain–machine interface and human–robot interaction. He received his MS degree in Computer Science from the Tokyo Institute of Technology, in 1986. In 1987, he joined Honda Research and Development Co., Ltd, and was engaged in researching intelligent assistance systems for cars, image understanding systems and brain-inspired reasoning systems. In 2003, he joined the Honda Research Institute Japan when it was established. His research focuses on creating intelligent machines that have interacting intelligence with humans in real-world situations. He is a Member of the IEEE, INNS, SFN, JSAI, JSSST and RSJ.