Autonomous Robots manuscript No.
(will be inserted by the editor)

The ManyEars Open Framework

Microphone Array Open Software and Open Hardware System for Robotic Applications

François Grondin · Dominic Létourneau · François Ferland · Vincent Rousseau · François Michaud

Received: date / Accepted: date

Abstract ManyEars is an open framework for microphone array-based audio processing. It consists of a sound source localization, tracking and separation system that can provide an enhanced speaker signal for improved speech and sound recognition in real-world settings. ManyEars software framework is composed of a portable and modular C library, along with a graphical user interface for tuning the parameters and for real-time monitoring. This paper presents the integration of the ManyEars Library with Willow Garage's Robot Operating System (ROS). To facilitate the use of ManyEars on various robotic platforms, the paper also introduces the customized microphone board and sound card distributed as an open hardware solution for implementation of robotic audition systems.

Keywords open source · sound source localization · sound source separation · mobile robotics · USB sound card · open hardware · microphone array

1 Introduction

Autonomous robots must be able to perceive sounds from the environment in order to interact naturally with humans. Robots operate in noisy environments, and limitations are observed in such conditions when using only one or two microphones [34]. In that regard, a microphone array can enhance performances by allowing a robot to localize, track, and separate multiple sound sources.

François Grondin, Dominic Létourneau, François Ferland, Vincent Rousseau, François Michaud
IntRoLab – Intelligent, Interactive, Integrated Robotics Lab
Interdisciplinary Institute for Technological Innovation
Université de Sherbrooke, 3000, boul. de l'Université, Sherbrooke (QC) Canada J1K 0A5
Tel.: +1-819-821-8000
Fax: +1-819-821-7937
E-mail: {francois.grondin2, dominic.letourneau, francois.ferland, vincent.rousseau, francois.michaud}@usherbrooke.ca

Using an array of eight microphones, ManyEars [29–31] demonstrated that it can, simultaneously and in real-time, reliably localize and track up to four of the loudest sound sources in reverberant and noisy environments [30]. ManyEars can also reliably separate up to three sources in an adverse environment with a suitable signal-to-noise ratio improvement for speech recognition [36,38]. ManyEars needs at least four microphones to operate, and the number of microphones used influences the number of sources that can be processed. It has mostly been used with arrays of eight microphones, to match the maximum number of analog input channels on the sound cards used. ManyEars has been used on different platforms including Spartacus [17], SIG2 [38] and ASIMO [36], and as a pre-processing module for improved speech recognition [33,37,39,40]. Many components of ManyEars are also used in HARK (HRI-JP Audition for Robots with Kyoto University) [20], an open source real-time system that also integrates new localization techniques such as GEVD MUSIC (Generalized EigenValue Decomposition Multiple Signal Classification) and GSVD-MUSIC (Generalized Singular Value Decomposition Multiple Signal Classification) [19,21,24].

The first implementation of ManyEars and HARK both rely heavily on FlowDesigner [16,28], an open source data flow development environment used to build complex applications by combining small and reusable building blocks. To facilitate maintenance and portability, ManyEars is now implemented in C as a modular library, with no dependance on external libraries. The source code is available online [11] under the GNU

GPL license [9]. A Graphical User Interface (GUI) (also available online [13]) is used to display in real-time the tracked sound sources and to facilitate configuration and tuning of the parameters of the ManyEars library. This paper presents these implementations and their integration to Willow Garage's Robot Operating System (ROS) [26].

To make use of the ManyEars library, a computer, a sound card and microphones are required. ManyEars can be used with commercially available sound cards and microphones. However, commercial sound cards present limitations when used for embedded robotic applications: they are usually expensive; they have functionalities such as sound effects, integrated mixing, optical inputs/outputs, S/PDIF, MIDI, numerous analogs outputs, etc., which are not required for robot audition; they also require significant amount of power and size. The EAR sensor has been proposed as an alternative [2], but it remains large and has strong coupling with the software which runs on a Field-Programmable Gate Array (FPGA). With ManyEars, computations are done on an onboard computer not embedded to the sound card, to facilitate portability and maintenance. To this end, the paper also introduces the customized microphone acquisition board and a 8-input sound card distributed as an open hardware alternative for robotic audition systems.

The paper is organized as follows. Section 2 presents the revised implementation of ManyEars as an open source C library. Section 3 introduces the GUI and explains ManyEars' portability by presenting its integration to ROS. Section 4 describes ManyEars' open hardware components and section 5 presents test cases to illustrate the use of the implemented framework.

2 ManyEars

Figure 1 illustrates the software architecture of ManyEars Library. It is composed of five modules: Preprocessing, Localization, Tracking, Separation and Postprocessing. These modules receive inputs and generate data using the Microphones, Potential Sources, Tracked Sources, Separated Sources and Postfiltered Sources data structures. In the following subsections, each of the five modules is described, along with the equations explaining what is implemented in the code. The parameters provided were set empirically to be robust to environmental changes, unless mentioned otherwise. More detailed explanations and justifications of these equations and parameters are available in [30] (Preprocessing, Localization and Tracking) and in [32] (Separation and Postprocessing). Also note that in this section, the variables m, l and k stand for the microphone, the frame and the bin indexes, respectively.

2.1 Preprocessing Module

The Preprocessing Module uses a MicST (Microphone Signal Transform) data structure to transform the time-domain signal of each microphone (sampled at 48000 samples/sec) in weighted frequency frames, as shown in Figure 2. The Preprocessor function transforms the microphone signal in the time domain into many individual frames of $N = 1024$ samples. Each frame is multiplied by a power-complementary window and then transformed in the frequency domain with a Fast Fourier Transform (FFT), which leads to $X^l_m[k]$. The MCRA (Minimum Controlled Recursive Averaging) function is used to estimate the spectrum of the stationary noise $(\lambda^s)^l_m[k]$ during silence periods [3]. The frames are initialized with zero values at frame $l = 0$ in equation 1, and then updated recursively. The weighting factor $\zeta^l_m[k]$ is then computed at each frequency bin according to equations 2, 3, and 4. The variables $\xi^l_m[k]$ and $(\lambda^r)^l_m[k]$ respectively represent the estimation of the *a priori* Signal-to-Noise Ratio (SNR) and the reverberation estimation [6,7]. The parameter $\alpha_d = 0.1$ is the adaptation rate, $\gamma = 0.3$ the reverberation decay for the room, and $\delta = 1.0$ the level of reverberation. These parameters need to be adjusted to the environment.

$$\begin{array}{r} (\lambda^r)^0_m[k] = 0 \\ (\lambda^s)^0_m[k] = 0 \\ \xi^0_m[k] = 0 \\ \zeta^0_m[k] = 0 \end{array} \quad 0 \le m < M, 0 \le k < N \tag{1}$$

$$(\lambda^r)^l_m[k] = \gamma(\lambda^r)^{l-1}_m[k] + \frac{(1-\gamma)}{\delta}\left|\zeta^{l-1}_m[k]X^{l-1}_m[k]\right|^2 \tag{2}$$

$$\xi^l_m[k] = \frac{(1-\alpha_d)\left|\zeta^{l-1}_m[k]X^{l-1}_m[k]\right|^2 + \alpha_d\left|X^l_m[k]\right|^2}{(\lambda^r)^l_m[k] + (\lambda^s)^l_m[k]} \tag{3}$$

$$\zeta^l_m[k] = \frac{\xi^l_m[k]}{\xi^l_m[k]+1} \tag{4}$$

2.2 Localization Module

Figure 3 illustrates the block diagram of the Localization Module. The Microphones data structure contains the cartesian positions (in meters) for each microphone in relation to the center of the array. A uniform unit

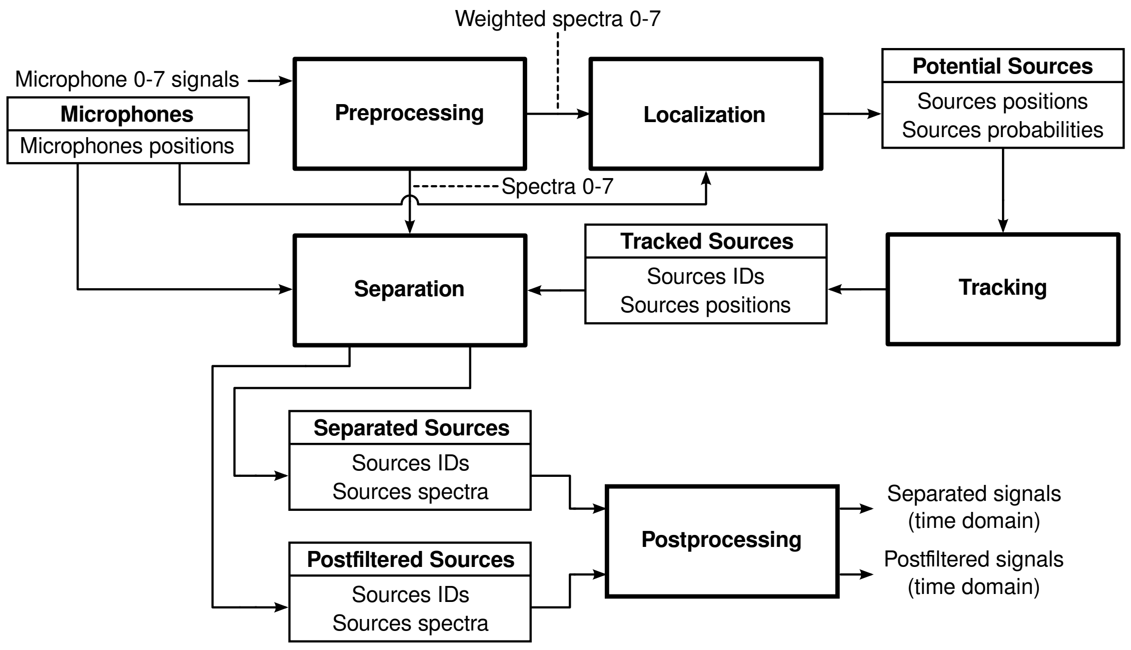


Fig. 1: Software Architecture of the ManyEars Library

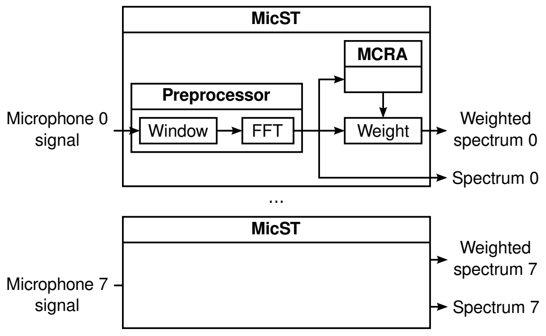


Fig. 2: Block diagram of the Preprocessing Module

sphere (with a 1 meter radius) is generated at the initialization of the Sphere data structure. This sphere is recursively generated from a tetrahedron, for a total of 2562 points. This resolution can be adjusted to satisfy real-time requirements. The delay between each pair of microphones for sound propagation of a source is precomputed at each point on the sphere during initialization of the Delays data structure, and stored in an array. Each delay corresponds to the direct path of sound, even if this hypothesis is influenced by the diffraction due to the body of the robot. However, experiments show that the system still performs well as long as a few microphones capture the direct path [30]. The cross-correlation $R^l_{m_1,m_2}(\tau)$ between microphones m_1 and m_2 is then computed for each new frames according to equation 5, with τ representing the delay.

$$R^l_{m_1,m_2}(\tau) = \sum_{k=0}^{N-1} \frac{\zeta^l_{m_1}[k]X^l_{m_1}[k]}{|X^l_{m_1}[k]|} \frac{\zeta^l_{m_2}[k]X^l_{m_2}[k]^*}{|X^l_{m_2}[k]|} e^{\left(\frac{j2\pi k\tau}{N}\right)} \tag{5}$$

To speed up computations, equation 5 is performed with an inverse Fourier Transform (IFFT). Although the IFFT reduces the number of operations, this step remains one of the most computationally expensive part of ManyEars. Moreover, since this operation is done for each pair of microphones, the complexity order is $O(M(M-1)/2)$, where M is the number of microphones[1]. Beamformer search is performed [30] and implemented in the Beamformer function. Once Q potential sources are found, their positions and probabilities are stored in the Potential Sources data structure. The position (x, y, z) of each potential source q is represented by the observation vector $\mathbf{O}^l_q$. The probability P^l_q for each potential source q to be a true source (and not a false detection) is computed according to equation 6. The variable E^l_0 stands for the energy of the beamformer for the first potential source, and the constant $E_T = 600$ represents the energy threshold adjusted to the environment (to find a good trade-off between false and missed sources detections). Experiments showed that the energy of the first potential source is related to the confidence that this is a valid source, while this is

[1] Set by parameter GLOBAL_MICSNUMBER in the file parameters.h

not the case for the next potential sources [30]. For this reason, the probability depends on the energy for $q = 0$ and is then associated to a constant value found empirically for the other sources ($0 < q < Q$). The probability for the first source is null when the energy is null, and goes to one as the energy goes to infinity. Moreover, it is relevant to notice that these probabilities are independent ($\sum_{p=0}^{Q-1} P_q^l \not\equiv 1$, $0 \leq E_0^l < \infty$).

$$P_q^l = \begin{cases} (E_0^l/E_T)^2/2, & q = 0, E_0^l \leq E_T \\ 1 - (E_0^l/E_T)^{-2}/2 & q = 0, E_0^l > E_T \\ 0.3 & q = 1 \\ 0.16 & q = 2 \\ 0.03 & q = 3 \end{cases} \tag{6}$$

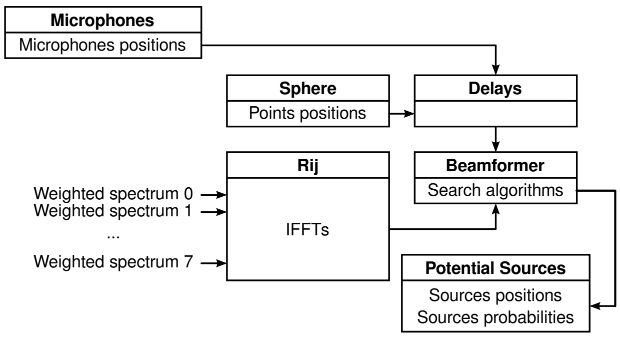


Fig. 3: Block diagram of the Localization Module

2.3 Tracking Module

Figure 4 represents the block diagram of the Tracking Module. There is a particle filter for each tracked source, represented by the Filter functions. There are S tracked sources and filters, each made of F particles. Each tracked source is assigned a unique ID and a position. The ID of a source stays the same over time as long as the source is active.

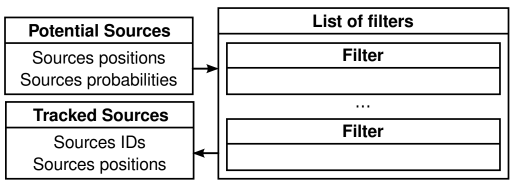


Fig. 4: Block diagram of the Tracking Module

The state vector of each particle, $\mathbf{s}_s^l(f) = [(\mathbf{x}_s^l(f))^T (\dot{\mathbf{x}}_s^l(f))^T]^T$, is composed of a (x, y, z) position, $\mathbf{x}_s^l(f)$, and a velocity, $\dot{\mathbf{x}}_s^l(f)$, where $(.)^T$ denotes the transpose operator. The beamformer provides this module with an observation vector for each frame l and potential source q, denoted by the variable $\mathbf{O}_q^l$. These observation vectors are concatenated in a single vector $\mathbf{O}^l = [\mathbf{O}_0^l, \ldots, \mathbf{O}_{Q-1}^l]$. Moreover, the vector $\mathbf{O}^{1:l} = \{\mathbf{O}^i, i = 1, \ldots, l\}$ stands for the set of all observations over time from frame 1 to frame l.

Table 1: Particle parameters

State	$\alpha_s^l(f)$	$\beta_s^l(f)$	Ratio
Stationary	2	0.04	10%
Constant velocity	0.05	0.2	40%
Acceleration	0.5	0.2	50%

During prediction, the position $\mathbf{x}_s^l(f)$ and velocity $\dot{\mathbf{x}}_s^l(f)$ of each particle f for source s are updated according to equations 7 and 8. The parameters $a_s^l(f)$ and $b_s^l(f)$ stand for the damping and the excitation terms, respectively. They are obtained with equations 9 and 10. The parameters $\alpha_s^l(f)$ and $\beta_s^l(f)$ are chosen according to the state of each particle. These values and the proportion of particles associated to each state are provided in Table 1. The parameter $\Delta T = 0.04$ stands for the time interval between updates. These three parameters are set to optimize tracking for both static and moving sound sources, and are robust to environmental changes since the source dynamics is independent of reverberation and noise. The variable F_x represents a normally distributed random variable.

$$\mathbf{x}_s^l(f) = \mathbf{x}_s^{l-1}(f) + \Delta T \dot{\mathbf{x}}_s^l(f) \tag{7}$$

$$\dot{\mathbf{x}}_s^l(f) = a_s^l(f)\dot{\mathbf{x}}_s^{l-1}(f) + b_s^l(f)F_x \tag{8}$$

$$a_s^l(f) = e^{-\alpha_s^l(f)\Delta T} \tag{9}$$

$$b_s^l(f) = \beta_s^l(f)\sqrt{1 - a_s^l(f)^2} \tag{10}$$

The position of each particle is normalized such that each particle stays on the unit sphere. The velocity is also normalized to ensure it is tangent to the sphere surface.

Each observation $\mathbf{O}_q^l$ is either a false detection (hypothesis $H0$), a new source not yet being tracked (hypothesis $H2$) or matches one of the sources currently tracked (hypothesis $H1$). The function $g_c^l(q)$ showed in equation 11 maps each observation $\mathbf{O}_q^l$ to a hypothesis.

The vector $\mathbf{g}_c^l$ introduced in equation 12 concatenates the mapping functions of all observations in a vector.

$$g_c^l(q) = \begin{cases} -2, H0: \mathbf{O}_q^l \text{ is a false detection} \\ -1, H2: \mathbf{O}_q^l \text{ is a new source} \\ 0, H1: \mathbf{O}_q^l \rightarrow \text{ source } s=0 \\ \vdots \\ S-1, H1: \mathbf{O}_q^l \rightarrow \text{ source } s=S-1 \end{cases} \quad (11)$$

$$\mathbf{g}_c^l = \{g_c^l(q), q = 0, \dots, Q-1\} \quad (12)$$

The variable c stands for the index of each realisation of the vector $\mathbf{g}_c^l$. There are $(S+2)^Q$ possible realisations, as demonstrated in equation 13.

$$\begin{array}{lllll} \mathbf{g}_0^l & = \{ & -2, \dots, & -2, & -2 \} \\ \mathbf{g}_1^l & = \{ & -2, \dots, & -2, & -1 \} \\ \mathbf{g}_2^l & = \{ & -2, \dots, & -2, & 0 \} \\ \mathbf{g}_3^l & = \{ & -2, \dots, & -2, & 1 \} \\ \vdots & & & \vdots & \\ \mathbf{g}_S^l & = \{ & -2, \dots, & -2, & S-2 \} \\ \mathbf{g}_{S+1}^l & = \{ & -2, \dots, & -2, & S-1 \} \\ \mathbf{g}_{S+2}^l & = \{ & -2, \dots, & -1, & -2 \} \\ \mathbf{g}_{S+3}^l & = \{ & -2, \dots, & -1, & -1 \} \\ \vdots & & & \vdots & \\ \mathbf{g}_{(S+2)^Q-2}^l & = \{ & S-1, \dots, & S-1, & S-2 \} \\ \mathbf{g}_{(S+2)^Q-1}^l & = \{ & S-1, \dots, & S-1, & S-1 \} \end{array} \quad (13)$$

The expression $P(\mathbf{g}_c^l|\mathbf{O}^{1:l})$ stands for the probability of a realisation $\mathbf{g}_c^l$ given the observations $\mathbf{O}^{1:l}$. Equation 14 introduces an alternative representation derived from Bayes inference.

$$P(\mathbf{g}_c^l|\mathbf{O}^{1:l}) = \frac{P(\mathbf{O}^{1:l}|\mathbf{g}_c^l)P(\mathbf{g}_c^l)}{\sum_{c=0}^{(S+2)^Q-1} P(\mathbf{O}^{1:l}|\mathbf{g}_c^l)P(\mathbf{g}_c^l)} \quad (14)$$

Conditional independence is assumed for the observations given the mapping function ($P(\mathbf{O}^{1:l}|\mathbf{g}_c^l)$), which leads to the decomposition expressed by equation 15. Independence of mapping functions is also assumed, and therefore the *a priori* probability $P(\mathbf{g}_c^l)$ is decomposed as shown in equation 16.

$$P(\mathbf{O}^{1:l}|\mathbf{g}_c^l) = \prod_{q=0}^{Q-1} p(\mathbf{O}_q^{1:l}|g_c^l(q)) \quad (15)$$

$$P(\mathbf{g}_c^l) = \prod_{q=0}^{Q-1} p(g_c^l(q)) \quad (16)$$

The probability distribution of the observations given the hypothesis is uniform for a false detection or a new source, and depends on the previous weights of the particle filter ($p(\mathbf{x}_s^{l-1}(f)|\mathbf{O}_q^{1:l-1})$) and the probability density of an observation given each particle position ($p(\mathbf{O}_q^{1:l}|\mathbf{x}_s^l(f))$), as shown in equation 17.

$$p(\mathbf{O}_q^{1:l}|g_c^l(q)) = \begin{cases} 1/4\pi & g_c^l(q) = -2 \\ 1/4\pi & g_c^l(q) = -1 \\ \sum_{f=0}^{F-1} \begin{pmatrix} p(\mathbf{x}_{g_c^l(q)}^{l-1}(f)|\mathbf{O}_q^{1:l-1}) \times \\ p(\mathbf{O}_q^l|\mathbf{x}_{g_c^l(q)}^l(f)) \end{pmatrix} & 0 \leq g_c^l(q) < S \end{cases} \quad (17)$$

The *a priori* probability $p(g_c^l(q))$ shown in equation 18 depends on the *a priori* probabilities that a new source appears and that there is a false detection. These values are represented by the variables $P_{new} = 0.005$ and $P_{false} = 0.05$. The probabilities that a source is observable and is a true source are represented by the variables $P(Obs_s^l|\mathbf{O}^{1:l-1})$ and P_q^l, respectively.

$$p(g_c^l(q)) = \begin{cases} (1-P_q^l)P_{false} & g_c^l(q) = -2 \\ P_q^l P_{new} & g_c^l(q) = -1 \\ P_q^l P(Obs_s^l|\mathbf{O}^{1:l-1}) & 0 \leq g_c^l(q) < S \end{cases} \quad (18)$$

Equation 19 shows that, given the previous observations $\mathbf{O}^{1:l-1}$, the probability that the source s is observable ($P(Obs_s^l|\mathbf{O}^{1:l-1})$) depends on the probability that the source exists ($P(E_s^l|\mathbf{O}^{1:l-1})$) and is active ($P(A_s^l|\mathbf{O}^{1:l-1})$).

$$P(Obs_s^l|\mathbf{O}^{1:l-1}) = P(E_s^l|\mathbf{O}^{1:l-1})P(A_s^l|\mathbf{O}^{1:l-1}) \quad (19)$$

The probability that a source is active, $P(A_s^l|\mathbf{O}^{1:l-1})$, is obtained with a first order Markov process. The transition probabilities between states are given by the expressions $P(A_s^l|A_s^{l-1}) = 0.7$ and $P(A_s^l|\neg A_s^{l-1}) = 0.3$, which respectively represent the probability a source remains active and becomes active.

$$\begin{aligned} P(A_s^l|\mathbf{O}^{1:l-1}) = & P(A_s^l|A_s^{l-1})P(A_s^{l-1}|\mathbf{O}^{1:l-1}) + \\ & P(A_s^l|\neg A_s^{l-1})(1 - P(A_s^{l-1}|\mathbf{O}^{1:l-1})) \end{aligned} \quad (20)$$

The active and inactive states are assumed to be equiprobable, and therefore the probability of activity

$P(A_s^{l-1}|\mathbf{O}^{1:l-1})$ is obtained with Bayes'rule in equation 21.

$$P(A_s^{l-1}|\mathbf{O}^{1:l-1}) = \left(1 + \frac{(1-P(A_s^{l-1}|\mathbf{O}^{1:l-2}))(1-P(A_s^{l-1}|\mathbf{O}^{l-1}))}{P(A_s^{l-1}|\mathbf{O}^{1:l-2})P(A_s^{l-1}|\mathbf{O}^{l-1})}\right)^{-1} \quad (21)$$

Equation 22 introduces $P(A_s^{l-1}|\mathbf{O}^{l-1})$, which stands for the probability a source is active. This relation is introduced in equation 22 with parameters $P_b = 0.15$ and $P_m = 0.85$.

$$P(A_s^{l-1}|\mathbf{O}^{l-1}) = P_b + P_m P_s^{l-1} \quad (22)$$

The expression P_s^{l-1} stands for the probability that the tracked source s is observed, which is obtained from the sum of the probabilities that this source is assigned to each potential source q ($P_s^{l-1}(q)$), as expressed by equation 23.

$$P_s^{l-1} = \sum_{q=0}^{Q-1} P_s^{l-1}(q) \quad (23)$$

Setting the *a priori* probability a source exists but is not observed to be $P_0 = 0.5$, the probability the source exists, $P(E_s^l|\mathbf{O}^{1:l-1})$, is obtained with equation 24.

$$P(E_s^l|\mathbf{O}^{1:l-1}) = P_s^{l-1} + \frac{(1-P_s^{l-1})P_o P(E_s^{l-1}|\mathbf{O}^{1:l-2})}{1-(1-P_o)P(E_s^{l-1}|\mathbf{O}^{1:l-2})} \quad (24)$$

The expression $P(\mathbf{g}_c^l|\mathbf{O}^{1:l})$ derives the probabilities that a new source is observed ($P_{H_2}^l(q)$), the source s is observed ($P_s^l(q)$) and there is a false detection ($P_{H_0}^l(q)$), as shown in equations 25, 26 and 27. The expression $\delta_{x,y}$ stands for the Kronecker delta. These probabilities are normalized for each value of q.

$$P_{H_0}^l(q) = \sum_{c=0}^{C-1} \delta_{-2,g_c^l(q)} P(\mathbf{g}_c^l|\mathbf{O}^{1:l}) \quad (25)$$

$$P_s^l(q) = \sum_{c=0}^{C-1} \delta_{s,g_c^l(q)} P(\mathbf{g}_c^l|\mathbf{O}^{1:l}) \quad (26)$$

$$P_{H_2}^l(q) = \sum_{c=0}^{C-1} \delta_{-1,g_c^l(q)} P(\mathbf{g}_c^l|\mathbf{O}^{1:l}) \quad (27)$$

The weight of each particle f is given by the expression $p(\mathbf{x}_s^l(f)|\mathbf{O}^{1:l})$, and is obtained recursively with equation 28.

$$p(\mathbf{x}_s^l(f)|\mathbf{O}^{1:l}) = \frac{p(\mathbf{x}_s^l(f)|\mathbf{O}^l)p(\mathbf{x}_s^{l-1}(f)|\mathbf{O}^{1:l-1})}{\sum_{f=0}^{F-1} p(\mathbf{x}_s^l(f)|\mathbf{O}^l)p(\mathbf{x}_s^{l-1}(f)|\mathbf{O}^{1:l-1})} \quad (28)$$

The observations may or may not match the tracked sources. The event I_s^l occurs when the source s is observed at frame l. The probability of this event is equal to the expression P_s^l. The expression $p(\mathbf{x}_s^l(f)|\mathbf{O}^l)$ stands for the probability the observation $\mathbf{O}^l$ matches the particle $\mathbf{x}_s^l(f)$, and is obtained in equation 29.

$$p(\mathbf{x}_s^l(f)|\mathbf{O}^l) = p(\neg I_s^l)p(\mathbf{x}_s^l(f)|\mathbf{O}^l,\neg I_s^l) + p(I_s^l)p(\mathbf{x}_s^l(f)|\mathbf{O}^l, I_s^l) \quad (29)$$

When the event I_s^l does not occur, all particles have the same probability $(1/F)$ to match the observations. The probability that the particle f matches the observation $\mathbf{O}^l$ ($p(\mathbf{x}_s^l(f)|\mathbf{O}^l), I_s^l$) is obtained from the probability each potential source $\mathbf{O}_q^l$ matches the particle f ($p(\mathbf{O}_q^l|\mathbf{x}_s^l(f))$). The denominator is needed to normalize the expression, as shown in equation 30.

$$p(\mathbf{x}_s^l(f)|\mathbf{O}^l) = (1-P_s^l)(1/F) + P_s^l\left(\frac{\sum_{q=0}^{Q-1} P_s^l(q)p(\mathbf{O}_q^l|\mathbf{x}_s^l(f))}{\sum_{f=0}^{F-1}\sum_{q=0}^{Q-1} P_s^l(q)p(\mathbf{O}_q^l|\mathbf{x}_s^l(f))}\right) \quad (30)$$

The expression $p(\mathbf{O}_q^l|\mathbf{x}_s^l(f))$ is obtained with the sum of gaussians shown in equation 31, and the variable d stands for the distance between the particle and the observation, as shown in equation 32. The initial model is inspired from a gaussian distribution that matches the distribution of the potential sources obtained from the beamformer. The model is then tuned empiricially to fit more accurately the observations, generating the distribution in equation 31.

$$p(\mathbf{O}_q^l|\mathbf{x}_s^l(f)) = 0.8e^{-80d} + 0.18e^{-8d} + 0.02e^{-0.4d} \quad (31)$$

$$d = \left\|\mathbf{x}_s^l(f) - \mathbf{O}_q^l\right\| \quad (32)$$

The estimated position of the tracked source $(\mathbf{x_{trk}})_s^l$ is finally obtained with equation 33.

$$(\mathbf{x_{trk}})^l_s = \sum_{f=0}^{F-1} p(\mathbf{x}^l_s(f)|\mathbf{O}^{1:l})\mathbf{x}^l_s(f) \tag{33}$$

This estimated position is sent to the tracked source structure, along with the source ID. Resampling is required when the particle diversity is lower than a predefined level ($N_{min} = 0.7F$), as shown in equation 34.

$$\frac{1}{\sum_{f=0}^{F-1} (p(\mathbf{x}^l_s(f)|\mathbf{O}^{1:l}))^2} < N_{min} \tag{34}$$

A new source may be added if $P^l_{H_0}(q)$ exceeds a threshold (fixed to 0.5), and a new filter is then assigned to this source. Each new source is assigned an ID. A source s being tracked can also be deleted when it stays inactive for too long ($P^l_s < 0.5$ for $l = (l_{now} - 24) : 1 : l_{now}$, where l_{now} is the index of the current frame). All currently tracked sources and their respective IDs are stored in the Tracked Sources data structure.

2.4 Separation Module

Figure 5 illustrates the Separation Module block diagram. Geometric Source Separation (GSS) is performed with the unmixing matrix $\mathbf{W}^l[k]$ in the GSS function, as expressed by equation 35. This matrix is initialized with the information from the Tracked Sources and the Microphones. This matrix is then optimized using equations 36 and 37 in order to minimize the independence (J_1) and geometric (J_2) costs. The gradient is used as it is a fast-convergence and low-complexity minimization solution [25]. The matrices $\mathbf{I}$ and $\mathbf{A}$ stand for the identify matrix and the direct propagation delays matrix, respectively. The matrix $\mathbf{A}$ is defined in equation 38, and the variable $\tau^l_{m,s}$ stands for the delay in samples at frame l, when sound leaves source s and reaches microphone m.

$$\mathbf{Y}^l[k] = \mathbf{W}^l[k]\mathbf{X}^l[k] \tag{35}$$

$$\frac{\partial J_1(\mathbf{W}^l[k])}{\partial(\mathbf{W}^l)^*[k]} = 4\left(\mathbf{E}^l[k]\mathbf{W}^l[k]\mathbf{X}^l[k]\right)\mathbf{X}^l[k]^H \tag{36}$$

$$\frac{\partial J_2(\mathbf{W}^l[k])}{\partial(\mathbf{W}^l)^*[k]} = 2[\mathbf{W}^l[k]\mathbf{A}^l[k] - \mathbf{I}]\mathbf{A}^l[k]^H \tag{37}$$

$$\mathbf{A}^l[k] = \begin{bmatrix} e^{j2\pi k\tau^l_{0,0}} & \dots & e^{j2\pi k\tau^l_{0,s}} \\ \vdots & \ddots & \vdots \\ e^{j2\pi k\tau^l_{m,0}} & \dots & e^{j2\pi k\tau^l_{m,s}} \end{bmatrix} \tag{38}$$

Update is performed with equation 39. The variables $\lambda = 0.5$ and $\mu = 0.001$ stand for the regularization factor and the adaptation rate, respectively.

$$\mathbf{W}^{(l+1)}[k] = (1-\lambda\mu)\mathbf{W}^l[k] - \mu\left[\left\|\mathbf{R_{mm}}^l[k]\right\|^{-2}\frac{\partial J_1(\mathbf{W}^l[k])}{\partial(\mathbf{W}^l)^*[k]} + \frac{\partial J_2(\mathbf{W}^l[k])}{\partial(\mathbf{W}^l)^*[k]}\right] \tag{39}$$

The covariance matrix of the microphones $\mathbf{R_{mm}}^l[k]$, the covariance matrix of the separated sources $\mathbf{R_{ss}}^l[k]$ and the intermediate expression $\mathbf{E}^l[k]$ are defined in equations 40, 41 and 42. These covariance matrices are obtained with instantaneous estimations and thus greatly reduce the amount of computations required. This approximation is similar to the Least Mean Square adaptive filter [12]. The operator diag sets all nondiagonal terms to zero.

$$\mathbf{R_{mm}}^l[k] = \mathbf{X}^l[k]\mathbf{X}^l[k]^H \tag{40}$$

$$\mathbf{R_{ss}}^l[k] = \mathbf{Y}^l[k]\mathbf{Y}^l[k]^H \tag{41}$$

$$\mathbf{E}^l[k] = \mathbf{R_{ss}}^l[k] - \mathrm{diag}(\mathbf{R_{ss}}^l[k]) \tag{42}$$

The spectra of the separated sources and their corresponding IDs (the same as for the tracked sources) are defined in the Separated Sources data structure.

Post-filtering is then performed on the separated sources. A gain is applied to the separated signals, as expressed by equation 43. The gain is computed according to interference and stationary noise [32].

$$Z^l_s[k] = G^l_s[k]Y^l_s[k] \tag{43}$$

Moreover, this step requires a MCRA function for each separated source to estimate the stationary noise. The new spectra and their corresponding IDs (the same as for both the tracked and separated sources) are defined in the Postfiltered Sources data structure.

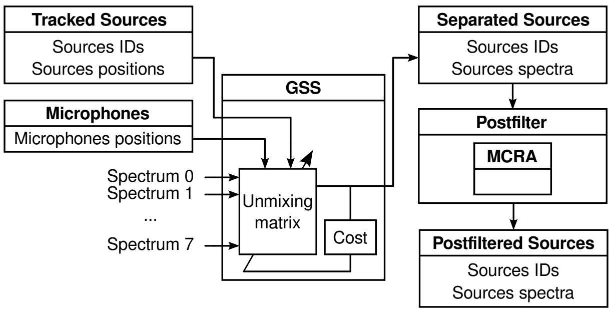


Fig. 5: Block diagram of the Separation Module

2.5 Postprocessing Module

As illustrated by Figure 6, the separated and postfiltered spectra from the Separated Sources and the Postfiltered Sources are then converted back to the time domain with IFFTs by the Postprocessor function. The new frames are then windowed and overlap-added to generate the new signals. Power-complementary windows are used for analysis and synthesis, and therefore overlap-add is required to achieve signal reconstruction.

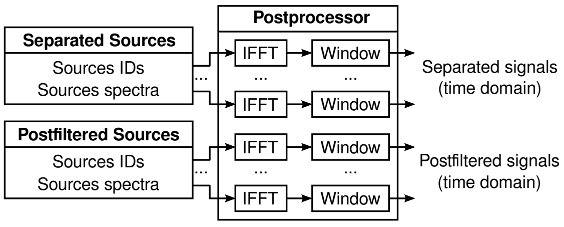


Fig. 6: Block diagram of the Postprocessing Module

3 The ManyEars Open Software Library Integrated to ROS

Implementation of ManyEars as an open software library with ROS involves translating the description of the architecture and algorithms presented in Section 2 into software processes, developing a Graphical User Interface (GUI) to visualize the tracked sound sources and to fine-tune the parameters of the ManyEars library, and interfacing the library to the ROS environment.

3.1 Software Processes

ManyEars' functions, as shown in the block diagrams of Section 2, are managed according to three stages: Initialization, Processing, and Termination. Figure 7 illustrates these stages. All the parameters are stored in the structure `parametersStruct`. This structure is used to provide parameters to the `functionStruct` during Initialization. Memory is also allocated for the elements of `functionStruct` during this step. This `functionStruct` is then used to perform many processing operations. During Processing, input arguments are used and output arguments are generated. Moreover, the elements of `functionStruct` are updated during this step. Finally, Termination is performed and the previously allocated memory is freed.

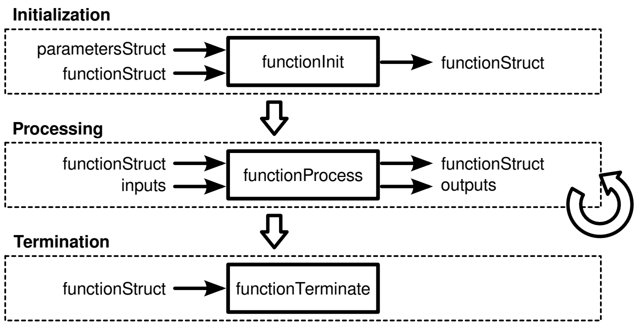


Fig. 7: Software structure of each module

To avoid dependency on external libraries, the following utility functions have been created and are used to perform various computations:

– Memory allocation. Memory is allocated to align arrays for SSE (Streaming SIMD Extensions) operations.
– FFT and IFFT. The FFT and IFFT operations are performed with a decimation-in-frequency radix-2 algorithm, which is optimized with SSE instructions. Moreover, since the signals are real in the time domain, two transforms are performed with each complex FFT or IFFT.
– Matrix operations. Most operations are performed on vectors and matrices. For this reason, customized functions for operations on vectors and matrices are created, and make use of SSE instructions.
– ID Manager. An ID manager is used to generate unique IDs to identify tracked, separated and postfiltered sources.
– Linear correlation. Linear correlation in the time domain is needed for the Postfiltering Module. This operation is also optimized with SSE instructions.
– Random number generator. Used by the Tracking Module, this function generates random numbers according to a uniform or a normal distributions.
– Transcendental function. This function estimates a confluent hypergeometric function for the Postfiltering Module.

– Window generation. This function generates a Hanning window for the Postfiltering Module. A power-complementary window is also generated for the Preprocessing Module and the Postprocessing Module.

3.2 Graphical User Interface

Figure 8 shows the GUI created for tuning parameters of the ManyEars Library. The GUI is a complementary tool not essential to use with the ManyEars Library. It consists of the following subwindows:

1. Microphone positions, beamformer configuration, source tracking and separation parameters. Parameters can be saved to and loaded from a file for rapid configuration.
2. Probabilities of sources calculated by the Localization Module in latitude.
3. Probabilities of sources calculated by the Localization Module in longitude.
4. Outputs of the Tracking Module in latitude.
5. Outputs of the Tracking Module in longitude.
6. 3D unit-sphere representation of the Tracked Sources.
7. Customizable colour representation of the information displayed by the GUI.

When the ManyEars GUI starts, the user selects to process the audio data either from a pre-recorded raw file or in real-time from the sound card. The application menu allows the user to start or stop processing and to select the audio input. Once the processing starts, subwindows (2) through (6) are updated as the audio input is being processed. The recorded data can also be saved to a raw file.

The GUI is implemented with the Qt4 framework [22] because of its flexibility, its open source licence and for the ability to create cross-platform applications.

3.3 ROS Integration

Figure 9 illustrates the integration of the ManyEars library with ROS [26]. Oval shapes represent ROS nodes, and rectangular shapes represent topics.

The integration with ROS is done with multiple simple nodes:

– `rt_audio_publisher`. This node publishes the raw audio data coming from the sound card in a ROS message called `AudioStream` containing the frame number and the stream data of all the microphones in 16 bits signed little endian format. The default publication topic is `/audio_stream`.

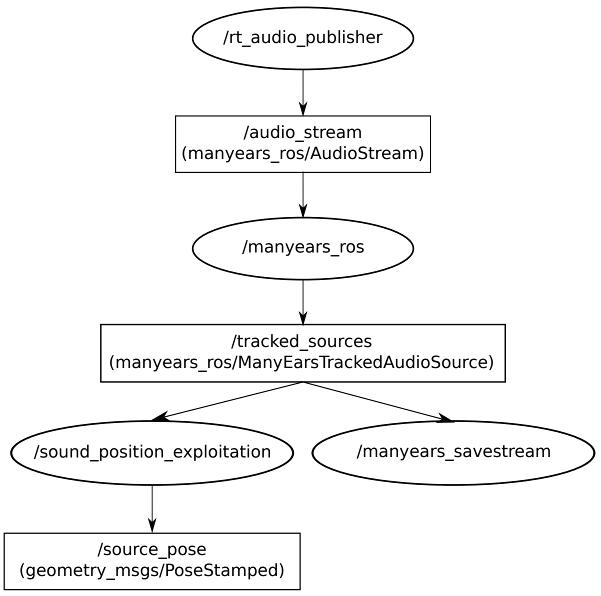
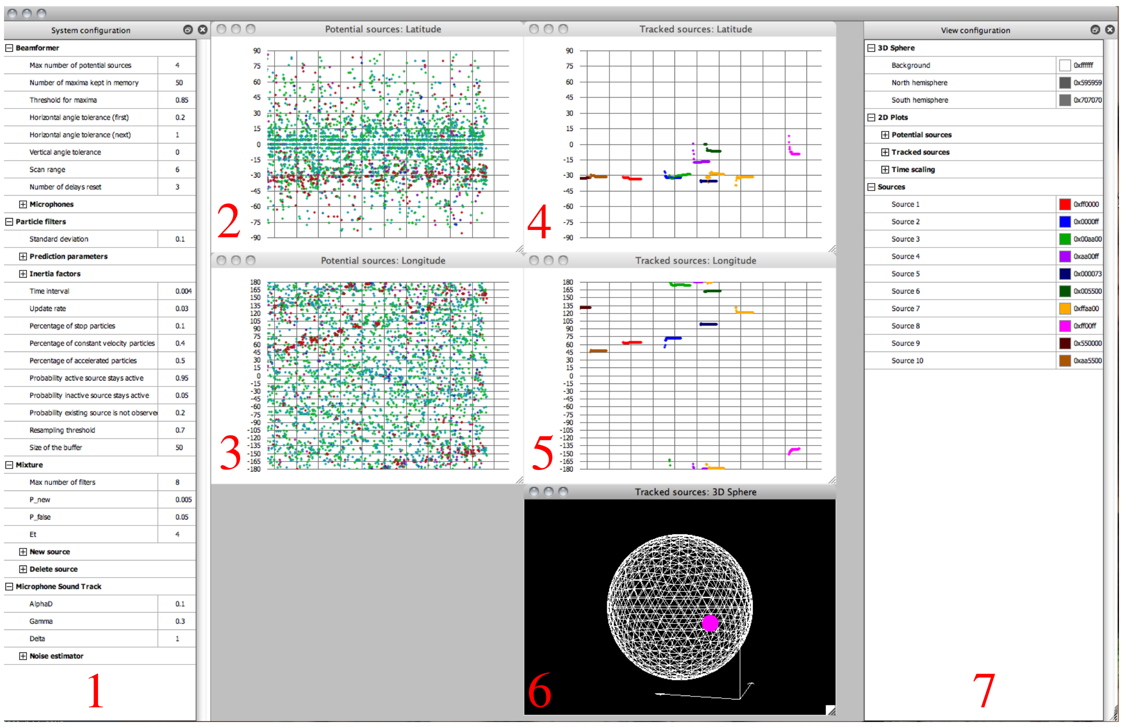

/rt_audio_publisher
/audio_stream
(manyears_ros/AudioStream)
/manyears_ros
/tracked_sources
(manyears_ros/ManyEarsTrackedAudioSource)
/sound_position_exploitation
/manyears_savestream
/source_pose
(geometry_msgs/PoseStamped)

Fig. 9: ManyEars-ROS nodes organization

– `manyears_ros`. This node uses the raw stream information published in `rt_audio_publisher` and executes the sound source localization, tracking and separation algorithm. It can use parameters saved by the ManyEars GUI described in Section 3.2. A message called `ManyEarsTrackedAudioSource` is published for each frame processed. This message contains an array of tracked sources of ROS message type `SourceInfo`. Each element of the array describes the source properties (ID, position, energy estimation, separation data, longitude, latitude). The default publication topic is `/tracked_sources`.
– `manyears_savestream`. This node connects to the `manyears_ros` node and uses the `separation_data` field in the `SourceInfo` ROS message to save the separated audio into WAV format files. This node can be used to listen to separated data. No ROS message is transmitted from this node. However, instead of saving to file the audio data in WAV format, the node could be easily modified to publish the audio data on a topic. This would be useful for other nodes that use single audio streams like speech recognition engines.
– `sound_position_exploitation`. This node is responsible for publishing pose information for each of the detected sound source. The node connects to the `tracked_sources` topic and outputs the right `geometry_msgs::PoseStamped` message with the correct orientation and unit distance from the center of geometry of the microphone array. The default publication topic is `source_pose`. Figure 10 shows


Potential sources: Latitude
Tracked sources: Latitude
Potential sources: Longitude
Tracked sources: Longitude
Tracked sources: 3D Sphere
1
2
3
4
5
6
7

System configuration

Beamformer	
Max number of potential sources	4
Number of maxima kept in memory	50
Threshold for maxima	0.85
Horizontal angle tolerance (first)	0.2
Horizontal angle tolerance (next)	1
Vertical angle tolerance	0
Scan range	6
Number of delays reset	3
Microphones	
Particle filters	
Standard deviation	0.1
Prediction parameters	
Inertia factors	
Time interval	0.004
Update rate	0.03
Percentage of stop particles	0.1
Percentage of constant velocity particles	0.4
Percentage of accelerated particles	0.5
Probability active source stays active	0.95
Probability inactive source stays active	0.05
Probability existing source is not observe	0.2
Resampling threshold	0.7
Size of the buffer	50
Mixture	
Max number of filters	8
P_new	0.005
P_false	0.05
Et	4
New source	
Delete source	
Microphone Sound Track	
AlphaD	0.1
Gamma	0.3
Delta	1
Noise estimator	

View configuration

3D Sphere	
Background	⬜ 0xffffff
North hemisphere	⬛ 0x595959
South hemisphere	⬛ 0x707070
2D Plots	
Potential sources	
Tracked sources	
Time scaling	
Sources	
Source 1	🟥 0xff0000
Source 2	🟦 0x0000ff
Source 3	🟩 0x00aa00
Source 4	🟪 0xaa00ff
Source 5	🟦 0x000073
Source 6	🟩 0x005500
Source 7	🟧 0xffaa00
Source 8	🟪 0xff00ff
Source 9	🟫 0x550000
Source 10	🟫 0xaa5500

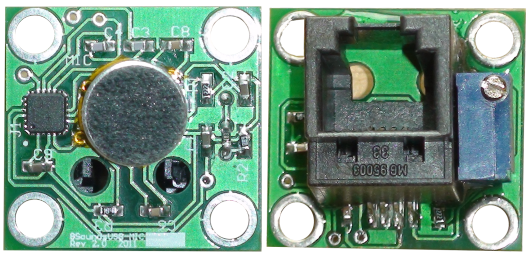
Fig. 8: ManyEars GUI

the result of the position data published with this node and visualized with the ROS RViz application.

4 Open Hardware Microphone Acquisition Board and 8-Input Sound Card

The design of a microphone acquisition board and a 8-input sound card specifically for ManyEars satisfy the following guidelines:

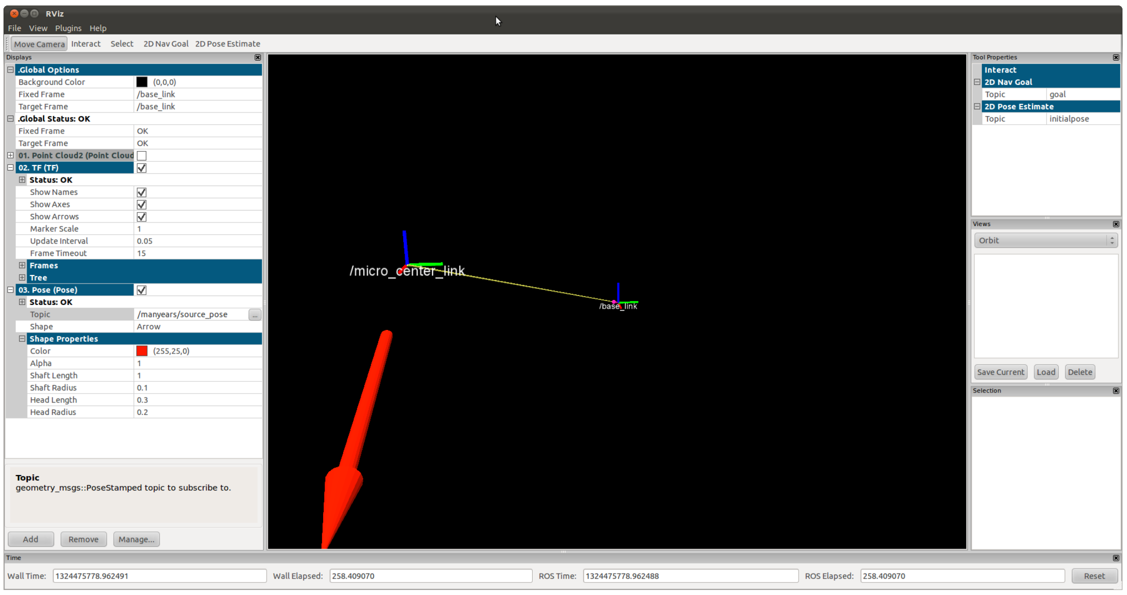
- Have small microphone boards powered by the sound card directly. Connectors must allow hot-plugging and must be low-cost. Installation of microphones must be easy.
- Minimalist design supporting up to eight microphone inputs and one stereo output.
- Minimize physical dimensions for installation on a mobile robot.
- Low power consumption and support of a wide range of power supply voltage.
- Minimum signal resolution of 12 effective bits and sampling rates from 8 to 192 kSamples/sec.
- Fabrication cost comparable or lower to commercially-available sound cards.
- Compatible with multiple operating systems (Linux, MacOS, Windows).
- Processing of the ManyEars algorithm is done externally, on the host computer, reducing processing power requirements on the sound card and facilitating portability and maintainability.



(a) Top (b) Back

Fig. 11: Microphone board

Figure 11 shows one of the microphone boards designed. Each microphone board has its own preamplifier circuit, which is powered by the sound card at 4.3 V. The main electronic components on the top side of the board include the omnidirectional microphone (CUI CMA-4544PF-W) and the preamplifier (STMicroelectronics TS472). The frequency response of the electret microphone is relatively flat from 20 Hz to 20 kHz. The back side of the board is composed of the RJ-11 connector (low-cost standard telephone jack style) and a potentiometer for easy preamplifier gain adjustment. Parallel insertion of the RJ-11 connector prevents the



.Global Options	
Background Color	■ (0,0,0)
Fixed Frame	/base_link
Target Frame	/base_link
.Global Status: OK	
Fixed Frame	OK
Target Frame	OK
01. Point Cloud2 (Point Cloud	☐
02. TF (TF)	✓
Status: OK	
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0.05
Frame Timeout	15
Frames	
Tree	
03. Pose (Pose)	✓
Status: OK	
Topic	/manyears/source_pose
Shape	Arrow
Shape Properties	
Color	🟥 (255,25,0)
Alpha	1
Shaft Length	1
Shaft Radius	0.1
Head Length	0.3
Head Radius	0.2

Topic
geometry_msgs::PoseStamped topic to subscribe to.

Interact	
2D Nav Goal	
Topic	goal
2D Pose Estimate	
Topic	initialpose

Wall Time: 1324475778.962491 Wall Elapsed: 258.409070 ROS Time: 1324475778.962488 ROS Elapsed: 258.409070

Fig. 10: Visualization of sound source position using ROS RViz

power line from making contact with the data line when insertion occurs, making the connection hot pluggable. Signals are mapped to the RJ-11 connector lines such that a standard telephone cable can be used. The connector also has a latch mechanism, which ensures reliable physical connections. The preamplifier has a high signal-to-noise ratio (more than 70 dB according to the TS472 datasheet), differential input and output channels and a maximum closed loop gain of approximately 40 dB, required to obtain a peak-to-peak amplitude of 4.3 V at the output. This maximizes the dynamic range of the codecs used by the sound card. Moreover, the microphone preamplifiers are positioned as close as possible to the electret microphone in order to reduce the effects of electromagnetic interference and to preserve a good signal-to-noise ratio.

To design the sound card, the first step was to choose the hardware interface to be used. The USB 2.0 High-Speed interface is a good choice because it is more commonly available compared to FireWire, and can be used directly to power the sound card unlike the standard Ethernet ports. The USB 2.0 transfer rate reaches 480 Mbits/sec, which is sufficient to transfer the raw (uncompressed) data of eight microphones and one stereo output. Recently introduced, the USB Audio Class 2.0 standard [15] includes more channels and better sampling resolutions and rates compared to Audio Class 1.0. The total consumption of the system must not exceed 2.5 W (500 mA @ 5V) for normal USB configuration. The design is based on the XMOS USB Audio 2.0 Multichannel Reference Design [35] to meet the power and interface requirements. This standard is convenient as it is automatically supported by standard drivers (ALSA for Linux, CoreAudio for OSX). On Windows platforms, a third party driver provided by XMOS partners is used because the USB Audio Class 2.0 is not yet supported natively. The XMOS is strictly used to operate the codec and forward the sound stream to the host computer (no processing is performed by the sound card).

Figure 12 shows a block diagram of the hardware implementation. The analog signal coming from the microphones is transmitted in differential mode to the codecs. Differential mode is preferred to single-ended signalling because of noise immunity and because it increases the dynamic range of the analog/digital converter of the codec. Since the sound card is designed to operate on a robot, many external devices can induce electromagnetic interference in the transmitted signal. Twisted pairs for differential signal transmission distribute the interference and a differential amplifier rejects the common mode noise, which minimizes the effect of overall electromagnetic interference.

The Preamp module uses a differential audio amplifier (National Semiconductor LME49726) that biases the signal for the codec's input and also filters the signal to avoid aliasing. This audio amplifier is especially intended for audio and it uses a single power supply. The band-pass filter in the Preamp module has a flat frequency response in the audio frequency band and has a rejection of 20 dB at the low frequencies. The configuration of the anti-aliasing filters is the one suggested by the codec manufacturer. Two four-input codec chips (Cirrus Logic CS42448) are used for analog to digital

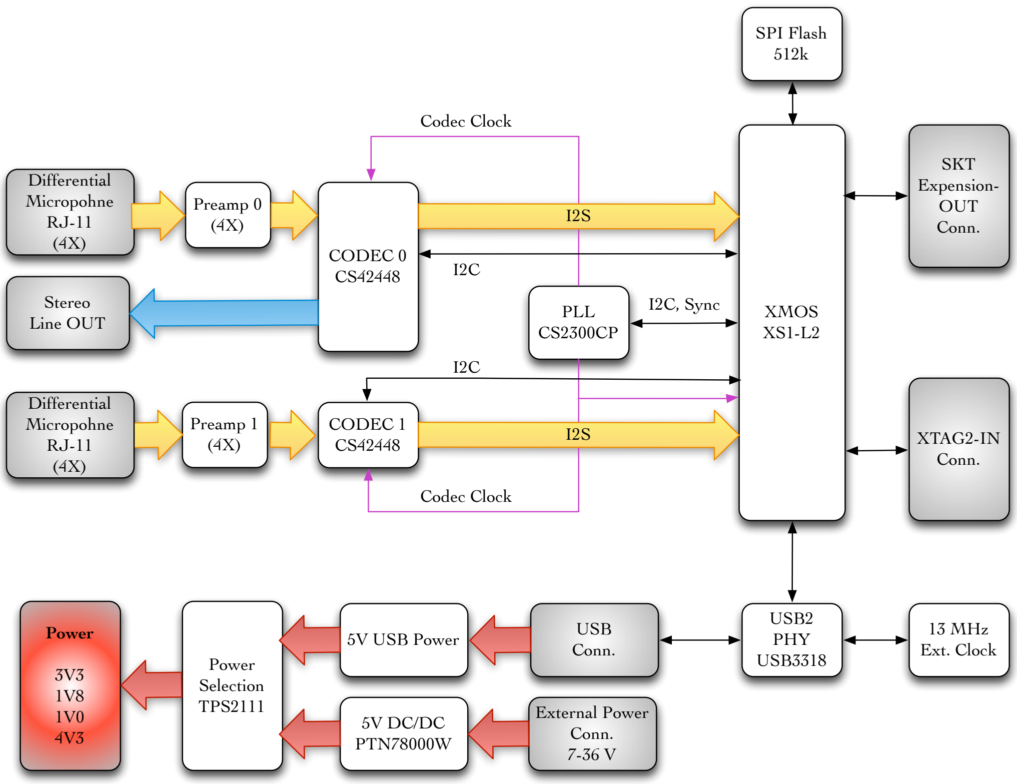


Fig. 12: Hardware Block Diagram

conversion. Data is transfered using the I2S protocol to the XMOS processor. The codecs are configured with the I2C protocol.

The XMOS XS1-L2 dual core microprocessor operates at 1000 MIPS and is of particular interest for mobile applications. Eight threads for each core are independently activated by events and do not run continuously with the system clock if not used. This reduces power consumption since only active threads require power. Threads are scheduled directly by the XMOS chip, making real-time performances possible without any operating system. An external PLL (Cirrus Logic CS2300CP) is used to synchronize the codecs and the XMOS cores. This is required to avoid jitter in the clock controlling the sampling of the analog inputs. The original XMOS firmware is used from the reference design, with a small addition to support the second codec. The firmware is stored in the 512k flash memory connected via SPI.

An XTAG2 connector (JTAG interface) is used for programming the SPI flash memory and for debugging. There is also an expansion port available compatible with XMOS standard SKT connector for future use. The XMOS processor is connected to the USB port using an external USB 2.0 PHY (SMSC USB3318). The PHY requires a 13 MHz clock to operate.

The sound card also has an external power connector (from 7 V to 36 V) if USB power is not available or insufficient. The switching power supply (Texas Instruments PTN78000W) uses the wide range input power and converts it effienciently to the required 5V. In case power is supplied both through USB and an external power supply, the Power Selection module (Texas Instruments TPS2111) prioritizes the external power source. Figure 13 presents a picture of the designed sound card.

Table 2 summarizes the characteristics of the designed microphone board and sound card. The microphone design files are available online [1] under the Creative Commons Attribution-ShareAlike 3.0 Unported license [4]. For the sound card, all the design files, gerbers and firmware are available online [1] under the Creative Commons Attribution-ShareAlike 3.0 Unported license [4].

5 Demonstration Test Cases

The ManyEars Library comes with many demonstration test cases available online [11], with parameters

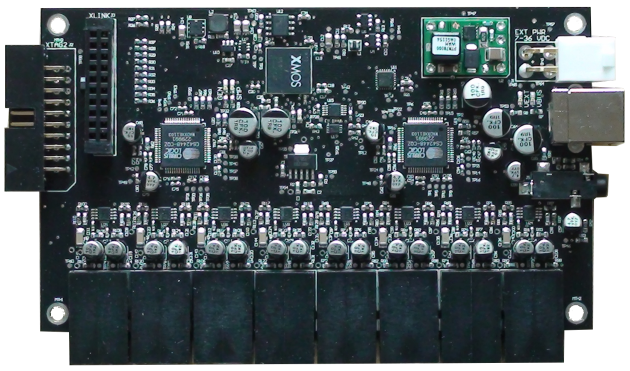


Fig. 13: Sound card

Table 2: Microphone Board and Sound Card Characteristics

Dimensions			
Item	Length (mm)	Width (mm)	Height (mm)
Microphone	23	23	15
Sound Card	125	74	15
Power Supply			
Power Source	Voltage (V)	Current (A)	Power (W)
External Power (without microphones)	7	0.332	2.324
	12	0.192	2.376
	24	0.114	2.736
	36	0.0884	3.182
External (with microphones)	7	0.343	2.401
	12	0.208	2.496
	24	0.118	2.843
	36	0.0884	3.182
USB (without microphones)	5	0.434	2.168
USB (with microphones)	5	0.449	2.246
Maximum with external power			3.182
Maximum with USB power			2.246
Additional Information			
Max. sampling latency between channels			16 us
Mean Noise Floor			-132 dBV
Maximum Noise Floor			-112 dBV

tuned to optimize performance. Two of these test cases are presented here: one with static sound sources, and another with moving sound sources. Figure 14 illustrates the coordinate system used in these test cases. The results in this section are displayed with a Matlab/Octave script (demo.m).

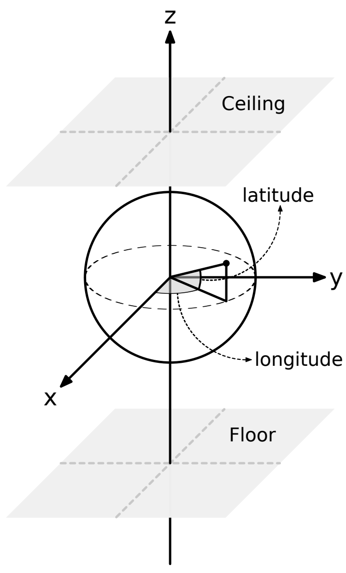
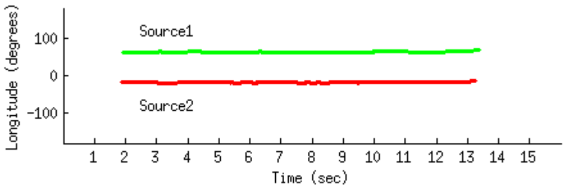


Fig. 14: Coordinate system

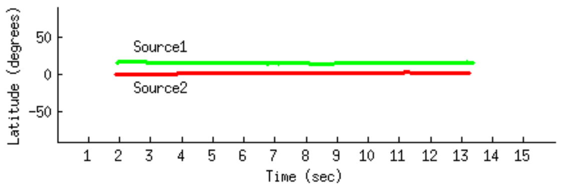
5.1 Static Sound Sources

This test case uses an 8-microphone cubic array. According to Rabinkin [27], the performance of a beamformer with speech sources (with a bandwidth between 100 Hz and 4 kHz) is optimized when the spacing between the microphones varies between 20 cm and 1 m. A 0.32 m × 0.32 m × 0.32 m array is used to make the spatial gain uniform and to fit on top of a mobile robot (e.g., Pioneer 2 platforms). The diameter of each microphone is 0.8 cm. For the results presented in this paper, the array is positioned at 0.6 m above the floor, in a room of 10 m × 10 m × 2.5 m with normal reverberation and some audible background noise generated by fans and other electronics. Two loud speakers with a diameter of 6 cm are used as sound sources, at a distance of 1.5 m and separated by 90°. Each speaker is placed approximately 0.6 m above the floor. Speech segments of two female speakers are played during 10 seconds. The signals of the microphones are recorded and then processed with the ManyEars Library.

Figure 15 represents the longitude and the latitude of the tracked sound sources. These positions match the locations of the loud speakers, with the small difference in latitude caused by the offset between the heights of the speakers. The localization error of ManyEars has been characterized to be less than 1° [29]. Figure 16 and Figure 17 show the spectrograms of source 1 and source 2, respectively. Separated and postfiltered spectrograms match many features in the clean spectrograms. Speech intelligibility and recognition are evaluated in [36–40].

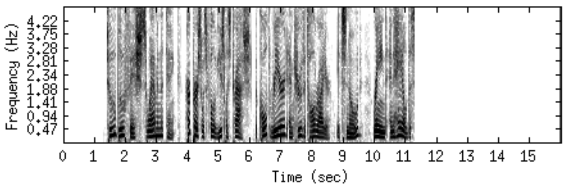


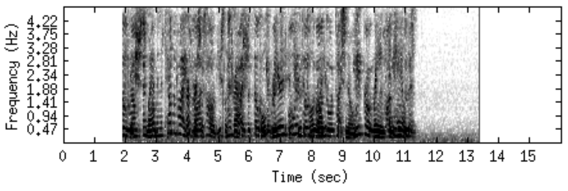
(a) Longitude



(b) Latitude

Fig. 15: Positions of the tracked sources


(a) Clean speech


(b) Separated speech

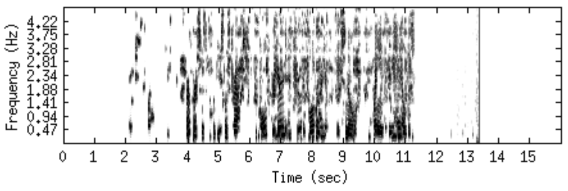
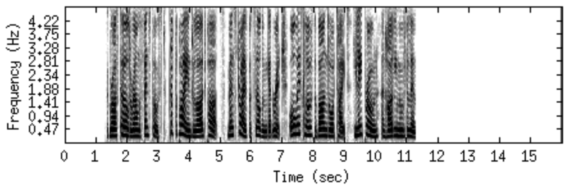
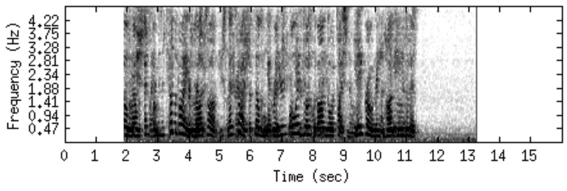

(c) Postfiltered speech

Fig. 16: Spectrograms for source 1


(a) Clean speech


(b) Separated speech

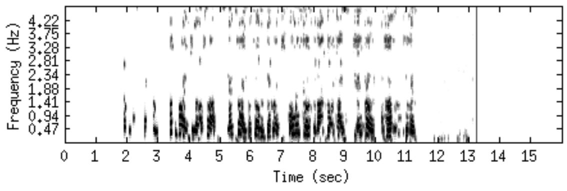

(c) Postfiltered speech

Fig. 17: Spectrograms for source 2

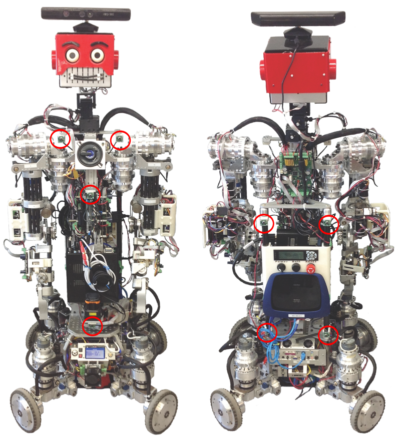


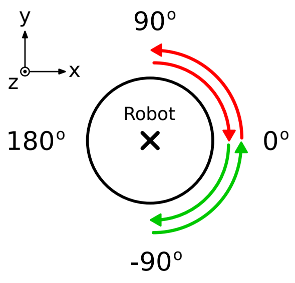
Fig. 18: IRL-1 with the microphones identified by the red circles

5.2 Moving Sound Sources

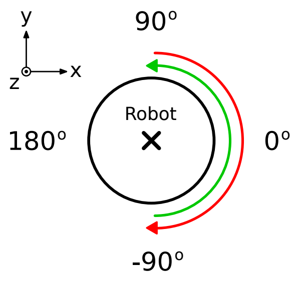
To demonstrate the use of ManyEars on a mobile robot and with an asymmetric array, this test case uses the microphones array on IRL-1, as shown by Figure 18. Two scenarios have been evaluated, illustrated by Figure 19, with human speakers producing uninterrupted speech sequences:

- Scenario A: The two sources are separated by 90° with respect to the xy-plane. They move by 90° and then come back to their initial position.
- Scenario B: The two sources are separated by 180° with respect to the xy-plane. They move to the position of the other speaker and cross each other.

As shown by Figure 20a, the tracked sources match the positions of the moving sources. In scenario B, the inertia of the particles used for tracking solve the source crossing problem. However, sources could have swapped if both speakers would have come close to each other at the same time, and then move back to their initial position. This problem can be solved by reducing the inertia of the particles (with parameters $\alpha_s^l(f)$ and $\beta_s^l(f)$ introduced in section 2.3), but then sources swapping could occur when speakers cross. Parameters of the Tracking module were tuned to find a trade-off between these two scenarios.

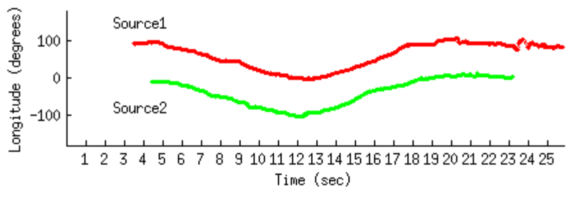


(a) Scenario A

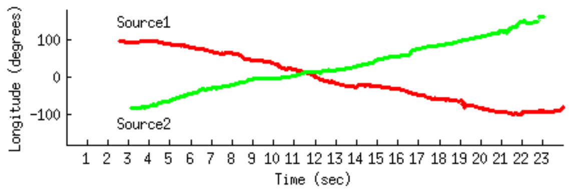


(b) Scenario B

Fig. 19: Positions of the moving sources



(a) Scenario A



(b) Scenario B

Fig. 20: Positions of the tracked sources

6 Conclusion

Compared to vision, there is not much hardware and software tools to implement and experiment with robot audition. The ManyEars Open Framework offers both software and hardware solutions to do so. The proposed system is versatile, portable and low-cost. The ManyEars C library is compatible with ROS and provides an easy-to-use GUI for tuning parameters and visualizing the results in real-time. Software and hardware components can be easily modified for efficient integration of new audio capabilities to robotic platforms. This new version of ManyEars has recently been used to demonstrate a speaker identification algorithm [10], and is currently used in augmented teleoperation and human-robot interaction scenarios with IRL-1, a humanoid robot with compliant actuators for motion and manipulation, artificial vision and audition, and facial expressions [8]. Integration of ManyEars and HARK libraries in ROS suggests that there is a potential for further standardization of ROS audio components, which could include data structures, standard DSP operations, audio codecs, and Matlab / Octave script integration. In addition, since the introduction of ManyEars, new methods have been proposed to detect the exact number of active sources [5,14], for tracking moving people [41], and for sound source separation using Independent Component Analysis [18], and these could be easily added to the ManyEars open framework. This effort would lead to a collection of useful, open source and portable tools similar to OpenCV [23] for image processing.

Acknowledgements This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation and the Canada Research Chair program.

References

1. Abran-Côté, D., Bandou, M., Béland, A., Cayer, G., Choquette, S., Gosselin, F., Robitaille, F., Telly Kizito, D., Grondin, F., Létourneau, D.: EightSoundUSB (2012). URL `http://eightsoundsusb.sourceforge.net`
2. Bonnal, J., Argentieri, S., Danes, P., Manhes, J.: Speaker Localization and Speech Extraction with the EAR sensor. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 670–675 (2009)
3. Cohen, I., Berdugo, B.: Noise Estimation by Minima Controlled Recursive Averaging for Robust Speech Enhancement. Signal Processing Letters **9**(1), 12–15 (2002)
4. Creative Commons: Attribution-ShareAlike 3.0 Unported (2012). URL `http://http://creativecommons.org/licenses/by-sa/3.0/legalcode`
5. Danes, P., Bonnal, J.: Information-Theoretic Detection of Broadband Sources in a Coherent Beamspace MUSIC Scheme. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 1976–1981 (2010)
6. Ephraim, Y., Malah, D.: Speech Enhancement Using a Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator. IEEE Transactions on Acoustics, Speech and Signal Processing **32**(6), 1109–1121 (1984)
7. Ephraim, Y., Malah, D.: Speech Enhancement Using a Minimum Mean-Square Error Log-Spectral Amplitude Estimator. IEEE Transactions on Acoustics, Speech and Signal Processing **33**(2), 443–445 (1985)

8. Ferland, F., Létourneau, D., Frémy, J., Legault, M.A., Lauria, M., Michaud, F.: Natural interaction design of a humanoid robot. To be published in the Journal of Human-Robot Interaction **1**(2) (2012)
9. Free Software Foundation, Inc.: GNU General Public License (2012). URL `http://www.gnu.org/licenses/gpl.html`
10. Grondin, F., Michaud, F.: WISS, a Speaker Identification System for Mobile Robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1817–1822 (2012)
11. Grondin, F., Valin, J.M., Létourneau, D.: The ManyEars Project: Microphone Array-Based Audition for Mobile Robots (2012). URL `http://manyears.sourceforge.net/`
12. Haykin, S.: Adaptive Filter Theory. Prentice Hall (2002)
13. IntRoLab: ManyEars ROS Package (2012). URL `http://introlab.github.com/introlab-ros-pkg/`
14. Ishi, C., Chatot, O., Ishiguro, H., Hagita, N.: Evaluation of a MUSIC-based Real-time Sound Localization of Multiple Sound Sources in Real Noisy Environments. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 2027–2032 (2009)
15. Knapen, G.: Universal Serial Bus Device Class Definition for Audio Devices (2006). URL `http://www.usb.org/developers/devclass_docs/Audio2.0_final.zip`
16. Létourneau, D., Valin, J.M., Côté, C., Michaud, F.: FlowDesigner: the Free Data-flow Oriented Development Environment. Software 2.0 **3** (2005)
17. Michaud, F., Côté, C., Létourneau, D., Brosseau, Y., Valin, J.M., Beaudry, E., Raievsky, C., Ponchon, A., Moisan, P., Lepage, P., Morin, Y., Gagnon, F., Giguere, P., Roux, M.A., Caron, S., Frenette, P., Kabanza, F.: Spartacus Attending the 2005 AAAI Conference. Autonomous Robots **22**(4), 369 – 383 (2007)
18. Mori, Y., Takatani, T., Saruwatari, H., Hiekata, T., Morita, T.: Blind Source Separation Combining SIMO-ICA and SIMO-Model-Based Binary Masking. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 81–84 (2006)
19. Nakadai, K., Ince, G., Nakamura, K., Nakajima, H.: Robot Audition for Dynamic Environments. In: Proceedings of the IEEE International Conference on Signal Processing, Communication and Computing, pp. 125–130 (2012)
20. Nakadai, K., Takahashi, T., Okuno, H., Nakajima, H., Hasegawa, Y., Tsujino, H.: Design and Implementation of Robot Audition System 'HARK' Open Source Software for Listening to Three Simultaneous Speakers. Advanced Robotics **5**(6), 739–761 (2010)
21. Nakamura, K., Nakadai, K., Ince, G.: Real-time Super-resolution Sound Source Localization for Robots. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 694–699 (2012)
22. Nokia corporation: Qt - A Cross-platform Application and UI Framework (2012). URL `http://qt-project.org/`
23. OpenCV: OpenCVWiki (2012). URL `http://opencv.willowgarage.com/wiki/`
24. Otsuka, T., Nakadai, K., Ogata, T., Okuno, H.G.: Bayesian Extension of MUSIC for Sound Source Localization and Tracking. In: Proceedings of the IEEE International Conference on Spoken Language Processing, pp. 3109–3112 (2011)
25. Parra, L., Alvino, C.: Geometric Source Separation: Merging Convolutive Source Separation with Geometric Beamforming. IEEE Transactions on Speech and Audio Processing **10**(6), 352–362 (2002)
26. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an Open-source Robot Operating System. In: Open-Source Software Workshop of the IEEE International Conference on Robotics and Automation (2009)
27. Rabinkin, D.: Optimum sensor placement for microphone arrays. Ph.D. thesis, State University of New Jersey (1998)
28. Valin, J.M., Létourneau, D.: Flow Designer (2008). URL `flowdesigner.sourceforge.net/`
29. Valin, J.M., Michaud, F., Rouat, J.: Robust 3D Localization and Tracking of Sound Sources using Beamforming and Particle Filtering. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 841–844 (2006)
30. Valin, J.M., Michaud, F., Rouat, J.: Robust Localization and Tracking of Simultaneous Moving Sound Sources using Beamforming and Particle Filtering. Robotics and Autonomous Systems **55**(3), 216–228 (2006)
31. Valin, J.M., Michaud, F., Rouat, J., Létourneau, D.: Robust Sound Source Localization using a Microphone Array on a Mobile Robot. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, vol. 2, pp. 1228–1233 (2003)
32. Valin, J.M., Rouat, J., Michaud, F.: Enhanced Robot Audition Based on Microphone Array Source Separation with Post-filter. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, vol. 3, pp. 2123–2128 (2004)
33. Valin, J.M., Yamamoto, S., Rouat, J., Michaud, F., Nakadai, K., Okuno, H.: Robust Recognition of Simultaneous Speech by a Mobile Robot. IEEE Transactions on Robotics **23**(4), 742 – 752 (2007)
34. Wolff, R., Lasseck, M., Hild, M., Vilarroya, O., Hadzibeganovic, T.: Towards Human-Like Production and Binaural Localization of Speech Sounds in Humanoid Robots. In: Proceedings of the IEEE International Conference on Bioinformatics and Biomedical Engineering, pp. 1–4 (2009)
35. XMOS ltd: USB Audio 2.0 Multichannel Reference Design (2012). URL `http://www.xmos.com/products/development-kits/usbaudio2mc`
36. Yamamoto, S., Nakadai, K., Nakano, M., Tsujino, H., Valin, J.M., Komatani, K., Ogata, T., Okuno, H.: Real-time Robot Audition System that Recognizes Simultaneous Speech in the Real World. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 5333–5338 (2006)
37. Yamamoto, S., Nakadai, K., Nakano, M., Tsujino, H., Valin, J.M., Komatani, K., Ogata, T., Okuno, H.: Design and Implementation of a Robot Audition System for Automatic Speech Recognition of Simultaneous Speech. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition & Understanding, pp. 111–116 (2007)
38. Yamamoto, S., Nakadai, K., Valin, J.M., Rouat, J., Michaud, F., Komatani, K., Ogata, T., Okuno, H.: Making a Robot Recognize Three Simultaneous Sentences in Real-time. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 4040–4045 (2005)
39. Yamamoto, S., Takeda, R., Nakadai, K., Nakano, M., Tsujino, H., Valin, J.M., Komatani, K., Ogata, T., Okuno, H.: Recognition of Simultaneous Speech by Estimating Reliability of Separated Signals for Robot Audition. Trends in Artificial Intelligence pp. 484–494 (2006)

40. Yamamoto, S., Valin, J.M., Nakadai, K., Rouat, J., Michaud, F., Ogata, T., Okuno, H.: Enhanced Robot Speech Recognition based on Microphone Array Source Separation and Missing Feature Theory. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1477–1482 (2005)
41. Yao, J., Odobez, J.M.: Multi-camera Multi-person 3D Space Tracking with MCMC in Surveillance Scenarios. In: Proceedings of the Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (2008)