Embedded Auditory System for Small Mobile Robots

Simon Brière, Jean-Marc Valin, François Michaud, Dominic Létourneau

***Abstract*— Auditory capabilities would allow small robots interacting with people to act according to vocal cues. In our recent work, we have demonstrated AUDIBLE, an auditory system capable of sound source localization, tracking and separation in real-time, using an array of eight microphones and running on a laptop computer. The system is able to localize and track up to four sources, while separating up to three sources in real-time in noisy environments. Signal processing techniques can be quite computer intensive, and the question of making it possible for this system to run on platforms that cannot carry a laptop computer onboard can be raised. This paper reports our investigation of the appropriate compromises to be made to AUDIBLE's implementation in order to port the system on an embedded DSP (Digital Signal Processor) platform. The DSP implementation is fully functional and performs well with minor limitations compared to the original system i.e., limitations on sound source duration and on the number of sources that can be processed simultaneously. Results demonstrate that it is feasible to port AUDIBLE on embedded platforms, opening up its use in field applications such as human-robot interaction in real life settings.**

I. INTRODUCTION

Localizing sound sources in our surroundings, or understanding somebody talking while moving in a crowd, are common in human interactions in real life. For a robot, however, such ability is not easily reproduced, having to deal with ambient noise and mixed sound sources. In the recent years, interest on artificial robotic audition has grown continuously, as it can be seen from the increasing number of robots exploiting such sense such as COG [1], SIG and SIG2 [2] and Spartacus [3], [4].

AUDIBLE is the name of the audition system used on Spartacus, developed to solve the problems of simultaneous sound sources localization, tracking and separation (SSLTS) [5], [6], [7], [8]. The system works in real-time using eight microphones, and is able to localize, track and separate simultaneous sound sources [9]. AUDIBLE was tested and demonstrated in various environments, such as the AAAI 2005 [3] and 2006 Mobile Robot competitions [10].

AUDIBLE is designed from ground up to run on a regular laptop, and requires most of its processing power. With limited processing capabilities on a robot, AUDIBLE takes up resources that cannot be used for other robotic tasks, such as vision. Adding a dedicated laptop requires space and energy, adds weight and increases cost, requirements that are not always easily met, especially for compact-size robots used for instance for vacuuming (e.g., Roomba from iRobot inc.) or to study human-robot interaction with autistic children or with toddlers [11]. Having the robot localize and track vocal cues would increase the interaction level with the persons involved. Separating multiple sound sources could provide cleaner audio stream to embedded speech recognition system (such as the Sensory Voice Direct II Toolkit), for improved performance. Our long-term objective is to have a compact, light, cheap and low power consumption SSLTS system to make such capabilities work on small mobile robots.

In this work, we investigate porting AUDIBLE on a DSP (Digital Signal Processor) board. The porting process is not straightforward, and design choices must be made affecting specific elements in AUDIBLE's implementation to allow the DSP version to work. The paper briefly explains the original system, putting in perspective the design choices required to build a functional embedded version of AUDIBLE. It also presents the design choices made when porting it to a DSP and the observed performance of these design choices. Finally, perspectives on how to improve this implementation are also outlined.

II. ORIGINAL AUDIBLE SYSTEM

The AUDIBLE system, illustrated in Fig. 1, is composed of a sound source localization subsystem that detects, localizes and tracks sound sources in the environment, and a sound source separation subsystem that uses the localization information to separate each source. The sampling rate used in the original system is 48 kHz. Speech recognition is not done by the system itself, but occurs at a subsequent stage. More specifically, AUDIBLE acts as a pre-processing module that provides sound source localization information and separated audio streams to other decisional modules.

A. Sound Source Localization

The sound source localization subsystem is described in [7], [9]. It consists of an initial localization step based on the steered response power algorithm and a tracking step that is performed using particle filtering. For the steered response power algorithm, the source direction is initially searched on a 2562-point spherical grid. The direction can be searched efficiently using only $N\left(N-1\right)/2$ sums per grid point :

$$direction = \underset{d}{\operatorname{argmax}} \sum_{i=0}^{N-1} \sum_{j=0}^{i-1} R_{i,j}\left(\mathrm{lookup}_{i,j}\left[d\right]\right) \qquad (1)$$

Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada, the Fonds Québecois de la Recherche sur la Nature et les Technologies, the Canada Research Chair program and the Canadian Foundation for Innovation.

S. Brière, D. Létourneau and F. Michaud are with the Department of Electrical Engineering and Computer Egineering, Université de Sherbrooke, 2500 boul. Université, Sherbrooke, Québec CANADA Simon.Briere@USherbrooke.ca Francois.Michaud@USherbrooke.ca Dominic.Letourneau@USherbrooke.ca

Jean-Marc-Valin is with the CSIRO ICT Centre, Sydney AUSTRALIA Jean-Marc.Valin@USherbrooke.ca

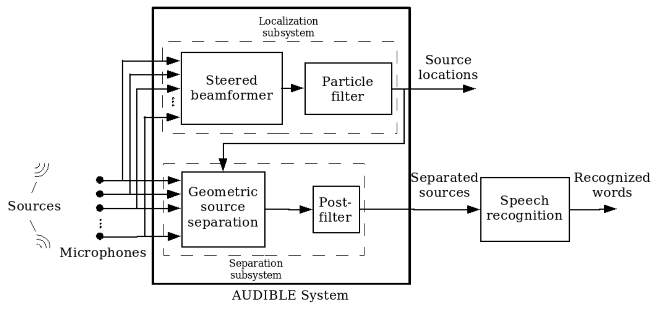

Fig. 1. Overview of AUDIBLE

where $\text{lookup}_{i,j}[d]$ is a lookup table that returns the time delay of arrival TDOA between microphones i and j for the searched direction d and $R_{i,j}$ is the relevance-weighted phase transform (RWPHAT) [7], [5], which is computed as:

$$R_{ij}(\tau) = \sum_{k=0}^{L-1} \frac{\zeta_i(k)X_i(k)\zeta_j(k)X_j(k)^*}{|X_i(k)|\,|X_j(k)|} e^{\jmath 2\pi k\tau/L} \qquad (2)$$

where $\zeta_i(k)$ is the Wiener gain for frequency k that takes into account both the noise and reverberation. Once a sound source is found using (1), it is possible to find subsequent sources, by forcing

$$R_{i,j}\left(\text{lookup}_{i,j}[direction]\right) = 0, \forall i, \forall j \qquad (3)$$

The search process is repeated to find a preset number of sources, which leads to false detections when fewer sources are present. The search in (1) is based on the far-field assumption (large distance to the array) with a grid that provides a maximum error of 2.5° (best case), which corresponds to the radius covered by each of the 2562 regions around its center.

It is however possible to improve the resolution by performing a refined search, constrained to the neighborhood of the first result found. In this second search, we can include the distance. While this distance estimate is not reliable enough to be useful, it helps improve the direction accuracy. In addition to the refining stage, most floor reflections can be eliminated by having the search exploit the fact that a reflection always has the same azimuth as the direct path, but with a higher absolute elevation.

The direction information found by the steered beamformer contains a large number of false positives and false negatives. Moreover, (1) is memoryless and is thus unable to keep track of sources over time, especially when there are *gaps* in the localization data for a source. For this reason, we use a particle filtering stage. The choice of particle filtering is motivated by the fact that taking into account false positives and false negatives makes error statistics depart significantly from the Gaussian model. Each source being tracked is assigned a particle filter and each observed direction in (1) is assigned to a tracked source using a probabilistic model [7]. By using the simple sample importance resampling (SIR) algorithm, it is possible to use 1000 particles per source while maintaining a reasonable complexity.

B. Sound Source Separation

The sound source separation subsystem [6], [9] is also composed of a linear sound source separation algorithm, followed by a non-linear post-filter. The initial linear source separation is achieved using a variant of the Geometric Source Separation (GSS) algorithm [12] that operates in real-time and with reduced complexity [6].

The GSS algorithm alone cannot completely attenuate the noise and interference from other sources, so a multi-source post-filter is used to improve the signals of interest. The post-filter is based on the short-term spectral amplitude estimator originally proposed by Ephraim and Malah [13]. Unlike the classical algorithm, the noise estimate used is the sum of two terms: stationary background noise and interference from other sources. The interference term is computed by assuming a constant leakage from the other sources [14].

III. EMBEDDING AUDIBLE ON A DSP

A. Hardware

The first task in porting the original system consists of selecting the embedded platform. Standard control processors (like PICs from Microchip) do not have enough computational power to process AUDIBLE's algorithm, and computer processors require a lot of electrical power to work. On the other hand, FPGA (Field-Programmable Gate Array) can be used to implement parallel algorithms, but it is hard to estimate the number of gates required for AUDIBLE, and the cost quickly increases with large number of gates. Therefore, the more promising option for this first embedded implementation is to use processors designed specifically for signal processing, i.e., DSP.

Because AUDIBLE's algorithm uses a lot of floating-point operations, we chose to use a floating-point DSP, more specifically the TMS320C6713 Texas Instruments DSP. The use of a fixed-point DSP is also possible, but would require more time to adapt the code for the processor. The TMS32C6713 is a 225 MHz floating-point processor with 256 kBytes of internal RAM memory with L1 and L2 cache support. According to the specifications, the processor is rated at 1800 MIPS and 1350 MFLOPS, and its architecture is optimized for audio processing, providing a bus to quickly transfer data between memory and external interfaces.

To capture the signals coming from the microphones, synchronized eight-channel analog-to-digital converters (ADC) are required to provide aligned audio frames. A communication interface is also required to transfer the processed data to a host system, typically a different computer on a robot. With all these considerations in mind, we chose to use a Lyrtech[1] CPA-II board .This board has 24 bits analog-to-digital converters supporting sampling frequencies from 32 kHz to 192 kHz. The board also provides 64 Mbytes of external memory (SDRAM, running at 100 MHz). It has a USB2 interface that provides the communication channel needed to transfer the processed data to the host system. The physical size of the CPA-II board is not an issue at

[1]http://www.lyrtech.com

this point, since we could design a smaller board once the software development on the DSP is completed.

B. Porting AUDIBLE on a DSP

The first step toward porting the original AUDIBLE implementation to the DSP is to convert the original C++ code into C code, which is better optimized by the DSP compiler. It is also necessary to remove dependencies to specialized libraries to carry out specific operations (e.g., FFTs), and find an equivalent way of implementing them on the DSP. Since the functions used in AUDIBLE are common in signal processing, this is done with a library included with the DSP.

The second step is to verify the accuracy of the code conversion. We use pre-recorded microphone signals that are injected in the DSP using an emulator. At various stages of the algorithm, the data coming out is validated to ensure it is the same as the data processed by the original system.

The last step is to optimize the code for real-time processing. In order to achieve real-time performance, a processing loop has to be under 10.66 ms (sampling at 48 kHz) or under 16 ms (sampling at 32 kHz). Optimization is done by using specific functions for the DSP and by modifying the loops to take advantage of the VLIW (Very Long Instruction Word) architecture that allows faster parallel calculations. At this stage, it becomes apparent that memory management is a critical element on the DSP. Internal memory is fast but limited, and external memory is slow but large. Since the algorithm uses a lot of tables (e.g., a 71736-bytes table is required to perform an accurate localization on a 2562 point grid around each microphone), it is impossible to fit all the code, the tables and the stack at the same time in the internal memory.

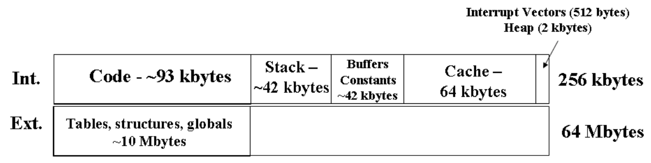
Interrupt Vectors (512 bytes)
Heap (2 kbytes)
Int.
Code - ~93 kbytes
Stack – ~42 kbytes
Buffers Constants ~42 kbytes
Cache – 64 kbytes
256 kbytes
Ext.
Tables, structures, globals ~10 Mbytes
64 Mbytes

Fig. 2. Memory mapping of AUDIBLE-DSP.

The memory mapping used is shown in Fig. 2. L2 cache (64 kbytes) is enabled to accelerate repeated external memory access. The memory section containing the program instructions (code section in the figure, around 93 kbytes) is placed in the internal memory for quick and repeated access. Because of the structure of the system, a large stack (around 42 kbytes) is used to allocate local variables. A section of the internal memory (around 42 kbytes) is reserved for general temporary buffers to speed some sections of the code. A small section of the internal memory is reserved for the interrupt vectors (512 bytes) and for the heap (2 kbytes). The external memory is mostly used as an audio buffer, large variables and for the large tables required by the algorithm, which currently uses around 10 Mbytes.

Because external memory is needed to store large tables that are accessed randomly – and thus cannot be properly cached – and because code optimization was done at the C level rather than at the assembly language level, the DSP implementation could not meet the same real-time performance of the original system, i.e., processing up to four sources at the same time with a sampling rate of 48 kHz. To allow the DSP implementation to process audio streams in real-time, the following modifications had to be made:

1) *Sampling rate:* In the original system, a sampling rate of 48 kHz was used. Using a 50% overlap for the separation subsystem and a frame size of 1024 samples, the processing is done in under 10.66 ms. In the DSP implementation, the sampling rate is lowered to 32 kHz, giving 16 ms for the maximum processing time between two 1024 samples frames with a 50% overlap.

2) *Number of localized and separated sources:* The number of localized and separated sources is brought down to 2 instead of the original value of 4.

3) *Directional refining:* In the original system, a direction refining process is done when a source is found, as described in [7] and in Section II. This requires extensive calculations, and has been removed from the DSP implementation.

4) *Particle filters:* The number of particles used in the particle filters is reduced empirically to 500 instead of the 1000 used in the original system.

5) *Buffering:* In order to keep up with real-time constraints even when the processing time is over 16 ms, we now use a super-frame technique that mainly consist of buffering frames and to process them when there is time. In the current implementation, a buffer of 200 frames is used. If, however, there is currently no sources being tracked and separated and the number of buffered frames gets over a threshold set to 25, the buffer is flushed. This is done in order to provide a good responsiveness of the system.

6) *Position refreshing:* In the original system, the positions of the sources were refreshed every 4 frames. This is a costly operation in terms of computational processing, and it is thus reduced to once every 5 frames on the DSP implementation.

These parameters are set empirically, because our objective for now is to evaluate feasibility. Work is currently underway to characterize in details the influence of each parameters of the different subsystems.

IV. RESULTS

To correctly rigorously evaluate the performance of our DSP implementation, we have to test each subsystem of AUDIBLE: localization, tracking and separation. We also have to collect information on the processing time of each subsystem in order to identify time-critical portions of the DSP implementation for future optimization. All tests are done using the original system parameters, with no optimization of the implementation's parameters for the specific test cases.

The experimental setup is shown in Fig. 3. Since some of the tests involve recorded sounds, an amplifier and two speakers positioned around the microphone array are used as sound sources. The microphone array is mounted on a cube, and each microphone is attached to one corner of the cube. Each microphone has a configurable gain. This gain is adjusted so that each microphone has the same amplitude with a given reference signal. The signal from each microphone is connected both to the ADC of the DSP board and to a capture card installed on a laptop.

Tests are conducted in a typical lab environment with people working as usual. No effort was done to reduce the background noise (ventilation system, chairs, computers, people and printer). Therefore, tests were conducted in noisy conditions, similar to what can be found in office-like environments. Laptop 1 runs the original AUDIBLE system, while Laptop 2 serves as a client system for the DSP. Laptop 2 is connected to the DSP using USB2. Each laptop records the reported sources position over time and each separated stream. Comparison of the two systems is possible because both are connected to the exact same microphone array. Each system having its own capture board, there is a different level of noise added in the signals during the sampling process. It is however assumed that this noise is negligible compared to environmental noise, making the differences observed between the original AUDIBLE system and the DSP system attributable only to the setup that produced the results.

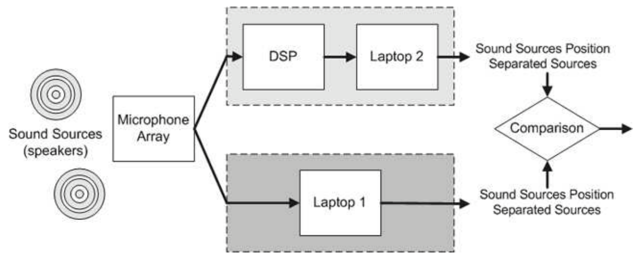
Sound Sources (speakers)
Microphone Array
DSP
Laptop 2
Sound Sources Position Separated Sources
Comparison
Laptop 1
Sound Sources Position Separated Sources

Fig. 3. Diagram of our experimental setup.

A. Processing Time

The first test on AUDIBLE-DSP measures processing time in different conditions. The timings are calculated using the internal DSP timer, averaged over a 5-second period. The results are shown in Table I. “Source” refers to a source being separated, while “filter” refers to a source being tracked. The Best Case time refers to the moment when the localization positions are not being refreshed (4 out of 5 frames). The Worst Case time refers to the moment when the positions are being refreshed (1 out of 5 frames). Some states are not possible and are not displayed in the table. The Idle time, t_{idle}, is defined as the amount of time the system is not doing anything over a 80 ms period (5 frames). The objective is to have a positive t_{idle}, since if the time is negative, the system has to buffer the frames for later processing, thus increasing the latency of the system. $t_{overflow}$, the maximum time before frame dropping occurs, can be calculated using (4):

$$t_{overflow} = \frac{Buf \cdot L \cdot t_{max}}{|t_{idle}| \cdot f_s} \quad (4)$$

where Buf is the buffer size (200 frames), L is the frame length (1024), t_{max} is the maximum time available to process a frame (16 ms) and f_s is the sampling rate (32 kHz).

TABLE I
PROCESSING TIME OF AUDIBLE-DSP

Status	Best Case (ms)	Worst Case (ms)	t_{idle} (ms)	$t_{overflow}$ (s)
0 source, 0 filter	8.5	25.6	20.4	∞
0 source, 1 filter	8.5	29.1	16.9	∞
0 source, 2 filters	8.5	33.3	12.7	∞
1 source, 1 filter	12.2	32.8	-1.6	64
1 source, 2 filters	12.2	37.0	-5.8	17.7
2 sources, 2 filters	15.0	39.8	-19.8	5.2

TABLE II
DETECTION RELIABILITY

Sound	Original system	DSP
Hand clap	100%	65%
Voice	100%	100%
Noise	100%	100%

According to these results, the DSP system is able to process 64 seconds of speech without frame dropping when there is one source being separated and tracked, but is only able to process 5.2 seconds while 2 sources are being separated and tracked. There is an increasing delay in the response of the system as the buffer is filling up, but the system is still able to operate in real-time. Negative Idle times indicate that the DSP implementation is dropping frames in these conditions, which may affect the quality of the separated streams and the precision of the position of the sound sources. Using a bigger frame buffer would delay the overflow, but would increase system latency.

Note that these times cannot be compared to the timing of the original system. Since that system runs on Linux (a non-real-time operating system), it is difficult to evaluate precise execution time of specific functions because there is no guarantee that a specific function will not be interrupted by the system scheduler.

B. Detection

Only one loudspeaker is used for this test. We consider sound source detection to be reliable if the system can detect every sound sources in its vicinity and if it is able to roughly localize it with a precision of 10° at distance of 1 meter. The loudspeaker is positioned on a 1-meter radius circle centered in the middle of the microphones array. The loudspeaker is placed at a height of 46 cm from the center of the microphone array, which is the origin of the positions reported by the localization system. The circle is divided into 16 equal sections of 22.5° each, starting at 0°. A pre-recorded audio stream consisting of 30 sounds is then played by the loudspeaker in each of the 16 positions on the circle. The audio stream is made of three types of sounds: hand clap, voice command (2 sec) and white noise burse (100 msec). Ten samples of these sounds occurring in sequence make the test stream.

Table II summarizes the results. The original system

TABLE III

LOCALIZATION ACCURACY DIFFERENCE BETWEEN THE DSP AND THE ORIGINAL SYSTEM

Sound	Azimuth	Elevation
Hand clap	-1.8°	-2.1°
Voice	-1.9°	-2.6°
Noise	-2.2°	-2.9°
Average	-2.0°	-2.6°

obtains a perfect score for the detection of each sound type. The DSP implementation also gets 100% detection for the voice and noise sounds, but does not perform as well with hand claps. This is caused by the position refreshing rate which is set at 1 every 5 frames for AUDIBLE-DSP. This statement has been verified by setting the refresh rate to the original value, bringing back a perfect score in detecting hand claps.

C. Localization

Using the same test setup of Section IV-B, two measures are taken in these trials to characterize AUDIBLE-DSP's localization capability: the accuracy of the azimuth angle of the detected sources, and the accuracy of their elevation. The root mean squared error is calculated by evaluating the difference between the angles returned by the DSP implementation and the original system.

The results shown in Table III represents the difference in localization accuracy between the DSP and the original system. On average, the DSP implementation is less accurate by $2.0°$ on azimuth and by $2.6°$ on elevation. The difference between the accuracy of the two systems comes mainly from the removal of the direction refining phase in the DSP implementation. By doing so, processing time is reduced, but so is accuracy. Considering that the original system accuracy is around $1.1°$ (azimuth) and around $0.89°$ (elevation) [9] in a similar environment, the global error of the DSP implementation can be estimated as $3.1°$ (azimuth) and $3.5°$ (elevation). Nonetheless, the accuracy obtained is sufficient for most applications and is similar to the human ear's accuracy [15], which ranges between $2°$ and $4°$ in similar conditions.

D. Tracking

In this test, instead of using a static loudspeaker, two persons are asked to walk at normal speed on a 2-meter radius circle around the microphones array, walking at normal speed and reading standard French text at normal pace. The tracking testing is done in two phases. In the first phase, the persons start at a precise position ($90°$ for the first person and $-90°$ for the second one), walk $90°$ to their right and then $180°$ to their left. This allows us to find the accuracy of the tracking in the case where the sound sources are not crossing. In the second phase, the persons start at a precise position ($180°$ and $0°$) and one walk $180°$ to the left and the other $180°$ to the right, crossing at $0°$.

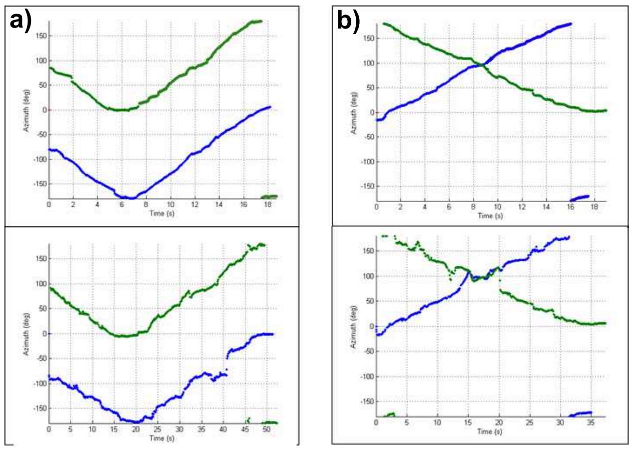

Fig. 4. Tracking results with two person. In a), the non-crossing path test, AUDIBLE to the top, AUDIBLE-DSP to the bottom. In b), the crossing path test, AUDIBLE system to the top, AUDIBLE-DSP to the bottom.

The resulting paths are shown in Fig. 4. Naturally, the trajectories tracked by the original system are smoother because the localization refresh rate is greater. At the crossing point, the DSP implementation also seems to confuse sound sources for a short time. These are probably caused by the reduction of the number of particles in the filters and the removal of the direction refining phase in the localization subsystem. However, even if the paths from the DSP implementation are less precise, the tracking is efficient because both speakers can clearly be tracked.

E. Separation

To characterize the separation subsystem, two fixed loudspeakers were placed at the following locations: $0°$ and $-90°$, $-90°$ and $135°$, $0°$ and $135°$. Three trials were conducted with a stereo recording made of 100 four-digit strings spoken by a mix of different speakers (half are female, half are male). We perform the tests using two sources of data: digits from the AURORA database [16] and recordings from volunteers. The original AUDIBLE localization subsystem is optimized for sampling rates over 20 kHz. Since separation is linked to the accuracy of the localization, samples from the AURORA databases (sampled at 8 kHz) are not well-suited to characterize the system, while the speech recordings from volunteers (sampled at 48 kHz) fits the optimization scheme of AUDIBLE.

In both case, the stereo stream is made of two simultaneous four-digits strings, one on the left channel and one on the right channel. The audio streams separated by AUDIBLE (original and DSP) are then sent to NUANCE automatic speech recognition (ASR) system[2] running on an external laptop. That way, the same ASR is used for both systems and the results can thus be compared. NUANCE's parameters were adjusted so that speech recognition accuracy on the individual digit strings (taken from AURORA and from volunteers) is 100%. Therefore, the speech recognition

[2]http://www.nuance.com

system is used here to assess the quality of the separation of AUDIBLE in its original and DSP implementations.

TABLE IV
RECOGNITION ACCURACY OF THE SEPARATION SUBSYSTEM

Tests	**Digit recognition rate**					
	Original			*DSP*		
	M	F	**Average**	M	F	**Average**
AURORA (8 kHz)	84%	80%	**82%**	83%	80%	**82%**
Volunteers (48 kHz)	95%	91%	**93%**	91%	88%	**90%**

Table IV shows the results of the separation subsystem (separation plus post-filtering modules). The recognition rate is calculated using each recognized digits in the strings, not strings as a whole. Results are compiled for both male (M) and female (F) voice recordings, and averaged over the two. Both the original and the DSP implementations work well with male and female voices, having at worst a 4% difference. With the 48 kHz samples corresponding to real life samples, the original system has an average 93% recognition rate and the DSP implementation has an average 90% recognition rate, which is still very good. In spite of the design compromises made, the separation performance of the DSP implementation is quite acceptable.

V. CONCLUSIONS AND FUTURE WORKS

By conducting this investigation on how AUDIBLE can be ported on a DSP implementation, this paper reports that such goal is feasible with acceptable localization, tracking and separation performances, by decreasing the sampling rate of the system to 32 kHz, using 500 particles for tracking with no direction refining, processing two sources simultaneously and using a super-frame technique to compensate for the limited internal memory on the embedded platform. AUDIBLE-DSP is capable to provide real-time localization, tracking and separation of short speech commands and audible cues. This study also contributes in outlining the influence of key elements of AUDIBLE's algorithm on localization, tracking and separation performances. The original AUDIBLE system was first designed with the objective of integrating the appropriate processing modules so that the system could work in real-time on a mobile robot operating in unconstrained conditions. While demonstrating that the system could be ported on an embedded platform to extend its usage to small robots, we also characterize the effect of specific elements of AUDIBLE's algorithm on its performance. Therefore, our paper also describes a methodology to conduct a comparative study of such auditory systems, with data that could benefit other comparative work.

Further works on the embedded system will aim at improving the performances of the system. Surely, a floating-point DSP with a larger internal memory would be beneficial. But now that we have a first embedded implementation, it may be worth investigating the combination DSP/FPGA, or even only the use of a FPGA, to improve processing speed and capabilities. Another option is to transfer the system on a fixed-point DSP to take advantage of lower power consumption, lower cost, higher internal clock and larger internal memory that such a processor provides. An embedded DSP solution (such as NUANCE's VoCon SF) could also be used for ASR. The main underlying objective of such improvements is to eventually come up with small, inexpensive and versatile auditory systems allowing to easily benefit from the advantages of hearing on all kinds of robots and systems operating in the real world.

REFERENCES

[1] R. Brooks, C. Breazeal, M. Marjanovie, B. Scassellat, and M. Williamson, “The Cog project: Building a humanoid robot,” *Computation for Metaphors, Analogy, and Agents*, vol. C. Nehaniv, Ed. Spriver-Verlag,, pp. 52–87, 1999.

[2] M. Murase, S. Yamamoto, J.-M. Valin, K. Nakadai, K. Yamada, K. K., T. Ogata, and H. G. Okuno, “Multiple moving speaker tracking by microphone array on mobile robot,” in *Proc. European Conf. on Speech Communication and Technology (Interspeech)*, 2005.

[3] F. Michaud, C. Côté, D. Letourneau, Y. Brosseau, J.-M. Valin, E. Beaudry, C. Raievsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguere, M.-A. Roux, S. Caron, P. Frenette, and F. Kabanza, “Spartacus attending the 2005 AAAI conference,” *Autonomous Robots (Springer)*, vol. 22(4), pp. 369–384, 2007.

[4] S. Brière, D. Létourneau, M. Fréchette, J.-M. Valin, and F. Michaud, “Embedded and integration audition for a mobile robot,” in *Proc. AAAI Fall Symposium Workshop Aurally Informed Performance: Integrating Machine Listening and Auditory Presentation in Robotic Systems*, vol. FS-06-01, 2006, pp. 6–10.

[5] J.-M. Valin, F. Michaud, and J. Rouat, “Robust 3D localization and tracking of sound sources using beamforming and particle filtering,” in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, 2006, pp. 221–224.

[6] J.-M. Valin, J. Rouat, and F. Michaud, “Enhanced robot audition based on microphone array source separation with post-filter,” in *Proc. IROS*, 2004.

[7] J.-M. Valin, F. Michaud, and J. Rouat, “Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering,” *Robotics and Autonomous Systems*, vol. 55, no. 3, pp. 216–228, 2007.

[8] J.-M. Valin, S. Yamamoto, J. Rouat, F. Michaud, K. Nakadai, and H. Okuno, “Robust recognition of simultaneous speech by a mobile robot,” *IEEE Trans. on Robotics*, vol. 22(4), pp. 742–752, 2007.

[9] J.-M. Valin, “Auditory system for a mobile robot,” Ph.D. dissertation, Université de Sherbrooke, 2005.

[10] F. Michaud, D. Létourneau, M. Frechette, E. Beaudry, and F. Kabanza, “Spartacus, scientific robot reporter,” in *Proc. AAAI Mobile Robot Workshop*, 2006.

[11] F. Michaud, T. Salter, A. Duquette, and J.-F. Laplante, “Perspectives on mobile robots used as tools for pediatric rehabilitation,” *Assistive Technologies, Special Issue on Intelligent Systems in Pediatric Rehabilitation*, vol. 19, pp. 14–29, 2007.

[12] L. C. Parra and C. V. Alvino, “Geometric source separation: Merging convolutive source separat ion with geometric beamforming,” *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 6, pp. 352–362, 2002.

[13] Y. Ephraim and D. Malah, “Speech enhancement using minimum mean-square error short-time spectral amplitude estimator,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[14] J.-M. Valin, J. Rouat, and F. Michaud, “Microphone array post-filter for separation of simultaneous non-stationary sources,” in *Proc. International Conf. on Acoustics, Speech, and Signal Processing*, 2004.

[15] B. Rakerd and W. M. Hartmann, “Localization of noise in a reverberant environment,” in *Proc. International Congress on Acoustics*, 2004.

[16] D. Pearce, “Developing the ETSI aurora advanced distributed speech recognition frontend & what next,” in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2001.