A USER-FRIENDLY SYSTEM FOR MICROPHONE ARRAY RESEARCH

Stuart E. Kirtman

The Cooper Union
for the Advancement of Science and Art
Department of Electrical Engineering
51 Astor Place
New York, NY 10003

Harvey F. Silverman

Laboratory for
Engineering Man/Machine Systems
Brown University
Division of Engineering
Providence, RI 02912

ABSTRACT

This paper describes the design and use of a digital acoustic array signal processing research environment. The system unites the interactive numerical processing capabilities of The MathWorks' MATLAB[1] with multichannel data acquisition / signal processing hardware and an overhead Cartesian robot for accurate sound source placement. In conjunction with additional electronic and electromechanical peripherals, the user-friendly environment facilitates the spatial and temporal analysis and evaluation of acoustic array algorithms. Real-data experiments and their results are provided to demonstrate the flexibility and ease-of-use of the array system.

1. INTRODUCTION

Microphone array technology promises to play a vital role in the non-invasive acquisition of high-quality acoustic data for speech recognition and teleconferencing technologies provided that robust algorithms can be developed to perform spatial filtering (beamforming) and source localization (tracking) under real-world conditions [1]. The development and testing of acoustic array algorithms in a real-life scenario can be extremely difficult and time consuming without specialized hardware and software tools that combine multichannel data acquisition/processing, advanced numerical computation, and the ability to position a sound source for spatial measurements. As with any system that promotes facile algorithm testing and development, it is imperative that all the features be accessible via a user-friendly interface and require a minimum investment in training time [2].

An environment fulfilling the above criteria has been developed and is depicted in Figure 1. The facility consists of a host workstation that provides overall control, an array processing unit (APU) that is responsible for collecting acoustic data from multiple microphones, an overhead robot capable of positioning an acoustic transducer at a user-specified X, Y, and θ coordinate, and assorted peripheral devices that extend the output, control, and measurement capabilities of the array. The requirements of powerful numerical processing and a user-friendly environment have been simultaneously satisfied by integrating the array hardware with

[1]MATLAB is a trademark of The MathWorks, Inc.

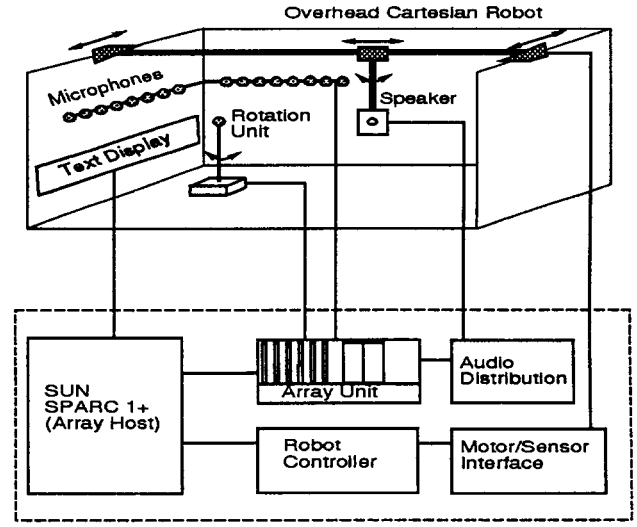

Figure 1: Microphone Array Environment

The MathWorks' MATLAB [3]. MATLAB offers advanced mathematical computation, extensive graphics capabilities, and supports a variety of "toolboxes" that supplement the environment and offer custom solutions to specific classes of problems. It has a powerful, easy-to-use interpreted programming language, is extensible with script and executable files, and has gained widespread acceptance in research, educational, and industrial institutions. The MATLAB user interface to the array environment consists of straightforward function calls that can be mastered in very little time. The remainder of this paper describes the array hardware and software, and examples are provided to illustrate the power and ease-of-use of this system.

2. ARRAY HARDWARE

The array has been designed to allow both off-line algorithm development and real-time signal processing of multichannel data. A number of subsystems comprise the hardware environment and include: a host workstation, a signal collection / processing unit, an overhead robot, a rotational unit, and miscellaneous peripherals. Figure 2 provides a

0-7803-2431-5/95 $4.00

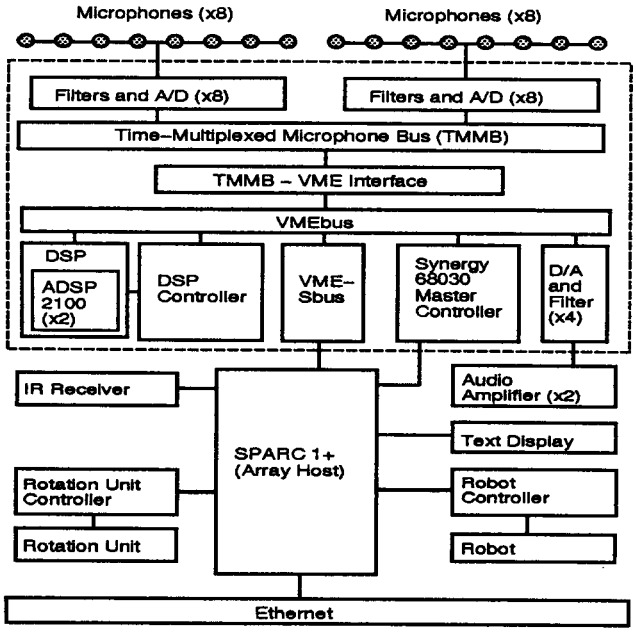

Figure 2: Hardware Block Diagram

block diagram of the hardware components and their interconnections; the following paragraphs describe each in greater detail.

There are numerous benefits to using an intelligent host as the center of a research system: the ability to use operating system facilities, video displays, keyboards, mass storage, advanced cross-development tools, availability of high-level languages, existing software applications, and access to additional resources via high-speed networks. For this application, a Sun SPARC 1+ workstation was chosen since it was readily available, fully integrated with additional computers, and supports a wide variety of high-quality software, including MATLAB. An Sbus/VMEbus parallel adaptor card was designed to provide a high-speed parallel interface to the array processing unit.

The array processing unit is the most specialized subsystem and was designed to provide a wide range of signal collection / processing functions. It is a 6U VMEbus-based system that can collect 12-bit digital acoustic data at a 20 kHz. or 16 kHz. sampling rate from up to 32 arbitrarily positioned microphones while synchronously emitting 12-bit digital audio via four analog output channels. There is sufficient memory to permit the storage of up to five seconds of data from 16 microphones at a 20 KHz. sampling rate, and the memory may be reapportioned as needed to accommodate specific tasks. Real-time processing of multichannel data (either for signal preconditioning or demonstration) is performed by two digital signal processors. Also included are a 16-character alphanumeric display, and a high-speed parallel link to the host.

An overhead robot has been designed to provide accurate, automated X, Y, and θ positioning of an acoustic transducer for testing and evaluating the spatial response of array algorithms. An AT-style personal computer interfaced to the array host controls stepper motor drivers that move a sound source throughout the test area. The robot is capable of accurately placing the transducer to within approximately $\pm$1cm over a 3.8m$\times$2.5m area and supports arbitrary velocity / acceleration for each degree of freedom.

To facilitate spatial acoustic measurements, it is often necessary to precisely rotate transducers. A small rotator unit (distinct from that on the overhead robot) has therefore been added to the array environment. A stepper motor driver half-steps a 48 steps/revolution stepper motor whose output shaft is easily coupled to a variety of transducers. The host controls the speed, direction, and other operating modes.

Since it is frequently useful to control certain features of the array while physically distanced from the keyboard (for example, when an arbitrarily located human talker initiates a multichannel recording), an infrared remote control has been added to the microphone array. Such an interface offers many advantages: the user is electrically isolated from the array, there are no cords to become tangled, and inexpensive multifunction commercial transmitters are readily available. Software has been written to distinguish the individual keypresses of the remote control for a typical VCR.

Just as it is useful to control the array from a distance, it is similarly valuable for the array to provide instructions or status to a remotely located individual. Therefore, a large, high-visibility, multi-color LED matrix display was incorporated into the array environment. Text can be displayed in a variety of colors, fonts, and scrolling patterns, and the unit may be programmed via an RS232C serial port interface, permitting direct interconnection with existing hardware.

3. ARRAY SOFTWARE

The host initializes the array processing unit and then communicates with the array using a well-defined server-client software model. A server running on the array processing unit responds to requests issued by a client program running on the host workstation, and the two exchange instructions and data following a simple protocol. With the current server, it is possible to vary the data collection time and number of input channels, play the recorded signal from any microphone, load up to two channels of digital data for analog playback during recording, initiate data collection/emission, and transfer multichannel data to the host.

The host client program (HCP) also provides an interface to the overhead robot, rotation unit, and infrared receiver, thus unifying all array functions within a single application. The HCP has a simple keyboard interface for manual operation, but has been enhanced to include the ability to control all of its features and transfer data directly to/from a remote machine (or process) using sockets, thus making it a socket-accessible server. Although the HCP must run on the host machine, any processor can connect to and use all the features of the array environment. While arbitrary programs supporting sockets can use the services of the array, for microphone array research, an ideal client is The MathWorks' MATLAB.

Support for sockets have been added to the MATLAB interpreter by using the "MEX" interface [4] that dynamically links C subroutines which have access to the full range of operating system I/O features. The MEX-files open a

socket connection to the server interface running on the host machine and provide the interpreter with the ability to send and receive data streams. MATLAB scripts ("M-files") spawn the array interface program on the host machine, serve as a front-end to the low-level MEX-files, and provide the necessary command formatting and error-checking. The scripts provide full support of all peripheral devices, including the robot and rotation unit.

4. EXAMPLE I: REAL-ROOM RESPONSE FOR INTEGER-DELAY DELAY-AND-SUM BEAMFORMER

The ease of performing experimental validation of array algorithms is demonstrated by obtaining the real-world spatial response of a delay-and-sum beamformer using an orthogonal linear array consisting of two eight-element subarrays with 16.5cm microphone spacing. The results were gathered by using the overhead Cartesian robot to place a speaker at 247 locations on an 1800mm×1200mm grid in a 3300mm×4800mm test area, transmitting an acoustic signal (a 500 to 5000 Hz. gated chirp) while recording the microphone signals, beamforming to an aiming point (1500mm, 1500mm), and then computing the average power. Figure 3 contains the simple MATLAB script to perform the

```
% Generate real-room delay-and-sum beamformer
% (integer delays) power plot
% Functions that begin with "serv_" control
% the array hardware
aimpt = [ 1.5 1.5 0]';
miclocs = getmics;
nsamps = 40000;
playsig = 1000*gen_chirp(500,5000,20000,nsamps);
delays = aim(aimpt, miclocs);
focus = makefocus(delays, nsamps);
serv_nsamps(nsamps);
serv_upld0(playsig);
T = -45;
idxY = 1;
for Y = 900: 100 : 2100
        idxX = 1;
        for X = 600 : 100 : 2400
                serv_rob_movsr(X, Y, T);
                pause(2);
                serv_cplay;
                data = serv_dnld;
                bf = beamform(focus, data);
                pwr(idxY,idxX) = (bf * bf');
                idxX = idxX + 1;
        end
        idxY = idxY + 1;
end
```

Figure 3: MATLAB script for delay-and-sum beamformer (integer delays)

task (some details such as the actual beamforming are in function calls and have not been included here due to space limitations), and Figure 4 shows resulting power plot. As anticipated, the beamformer provides a significant peak in spatial response at the aiming point. Also evident in the figure is the dominance of near-field effects as the sound

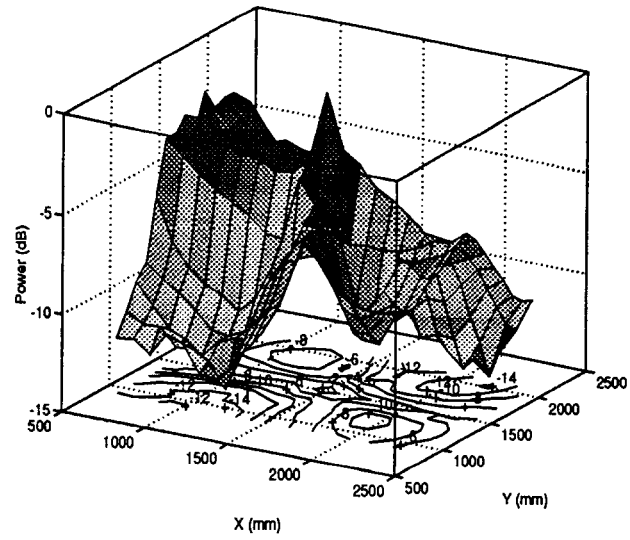

Figure 4: Real-room power plot for delay-and-sum beamformer (integer delays)

source approaches the array. Doing this experiment manually would be an *extremely* arduous task that would take many hours to complete. Using the array environment, this experiment required minimal manual labor and had a total running time of approximately 90 minutes.

5. EXAMPLE II: THE EVALUATION OF AN ACOUSTIC SOURCE LOCATION ALGORITHM

In a previous publication [5], a correlation-based algorithm was presented for establishing the two-dimensional (X-Y plane) location of an active sound source using a linear microphone array. Although real data was used in the performance evaluation, the time and effort required to collect data from a large number of spatial points made complete testing quite prohibitive. The user-friendly array research system described here addresses this need and has allowed a more thorough characterization of the location algorithm's performance.

For discrete (sampled) signals, the proposed method determines source position by maximizing a functional of the interpolated cross-correlations between microphones. The cross-correlations are interpolated to improve location accuracy by providing better resolution than that obtainable by using the data at the sampling rate. The amount by which the cross-correlation is interpolated depends upon the accuracy required; interpolation by a factor of 64 was used in this example. The location algorithm is summarized by the following:

1. Assume the sound source to be at a point (X, Y) and determine (using geometry and the appropriate sound wave propagation model) the source's time-difference of arrival (TDOA) for all of the independent microphone pairs (the microphone pairing may be selected arbitrarily).
2. For each microphone pair, convert the TDOA deter-

mined in (1) into a corresponding delay (lag) in the interpolated cross-correlation of the signals received by the microphones associated with that pair.

3. Using the real microphone data, for each of the pairs determine the value of the interpolated cross-correlation at the lag calculated in (2), i.e. the lag that corresponds to the TDOA for a signal located at (X, Y). Compute the sum of the interpolated cross-correlation values obtained from each of the independent pairs (call this value S).

4. Repeat (1-3) for all potential (X, Y) locations where the sound source might be located; the hypothesized point with the maximum sum-of-interpolated-cross-correlations, S, is the estimated sound source location.

In practice, it is not feasible to evaluate the functional at every potential source location and therefore for testing purposes, the region of potential source locations was divided into a uniformly-spaced 25mm×25mm grid, and the functional was evaluated at each grid-point. An alternative to this grid-search method is the application of a non-linear optimization technique such as Stochastic Region Contraction (SRC) [6]. This offers the possibility to significantly reduce the number of functional evaluations required to find the global maximum, and provides the ability to determine source locations that lie between grid-points.

In the algorithm originally published, a single linear array and a two-stage search method were used to find the X and Y coordinates of the source. Here, however, two *orthogonal* arrays – one along the X axis and one along the Y axis – are used to more accurately obtain source position. Like its predecessor, this newer method is a two-stage algorithm. In the first pass, the X array is used to determine the X coordinate of the source; the second pass uses the Y array to compute the Y coordinate of the source.

The locator algorithm was implemented with MATLAB scripts and was evaluated by using the overhead Cartesian robot to place a speaker at 196 points on a 1950mm×1950mm grid (150mm spacing between testing points), play half a second of speech from a male talker, collect the microphone data, and calculate an estimate of the speaker location. The locator used 2048 samples to perform the cross-correlation, interpolation by a factor of 64, and a searching grid of 25mm. The total running time was approximately 60 minutes.

Figure 5 shows magnitude of the error computed as $Error = \sqrt{(X_{est} - X_{real})^2 + (Y_{est} - Y_{real})^2}$. The performance is clearly superior to that reported previously in [5]; here, the majority of the estimates are within approximately 100mm of the correct value. Most of erroneous estimates occur when the sound source is located outside the array apertures. This is due, in part, to the directionality of the sound source and the changes in room characteristics as a function of source position. Moreover, the graph only indicates the combined X and Y errors and does not reflect directly reflect the individual X and Y errors.

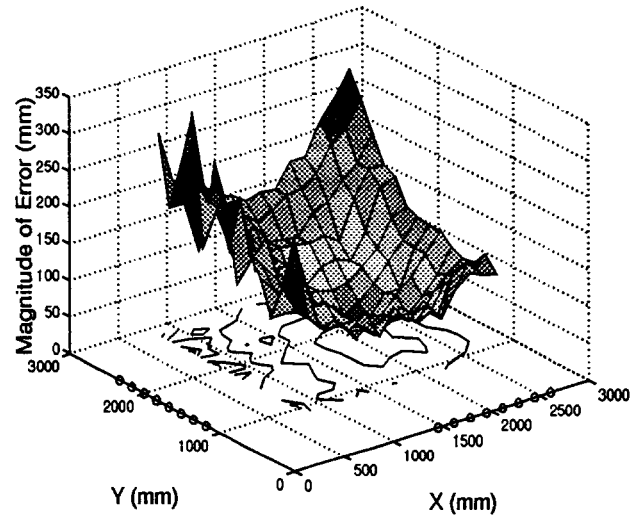

Figure 5: Magnitude of the error of the source location estimate

6. CONCLUSION

An environment has been described that provides a powerful, yet simple-to-use, mechanism for doing experimental research on microphone arrays. The two examples provided have demonstrated the user-friendly nature of the environment and the types of experiments that can be performed. While the creation of such a system is a large investment, it is proving to have been a sensible one; several new **real-data-tested** algorithms have been quickly improved and validated through its use. It has become evident that this type of environment is essential for meaningful experimental research in this area.

7. REFERENCES

[1] Flanagan, J.L., J.D. Johnston, R. Zahn, G.W. Elko, "Computer-steered microphone arrays for sound transduction in large rooms", *Journal of the Acoustical Society of America* vol. 78, November 1985.

[2] Russo, Fabrizio and Sandro Broili, "A user-friendly environment for the generation of highly portable software in compuer-based instrumentation", *IEEE Transactions on Instrumentation and Measurement*, Vol. 39, No. 6, December 1990.

[3] The MathWorks, Inc., *Matlab Reference Guide*, August 1992.

[4] The MathWorks, Inc., *Matlab External Interface Guide*, August 1992.

[5] Silverman, Harvey F. and Stuart E. Kirtman, "A two-stage algorithm for determining talker location from linear microphone array data", *Computer Speech and Language*, Vol. 6 (1992).

[6] Berger, M. and H.F. Silverman, "Microphone array optimization by stochastic region contraction (SRC)", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 39, No. 11, November 1991.