DISTRIBUTED MICROPHONE ARRAYS FOR DIGITAL HOME AND OFFICE

Ying Jia, Yu Luo, Yan Lin

Intel China Research Center, Intel Corporation
PRC

Igor Kozintsev

Intel Labs, Intel Corporation
USA

ABSTRACT

With the proliferation of pervasive computing in digital home and office environment there is an increasing demand for hands-free audio and video interfaces. Microphone arrays are already present on desktops and mobile computers, and camera arrays are in transition from being highly sophisticated prototypes to become affordable consumer devices soon. Thanks to increasing CPU power of general-purpose computers and higher bandwidth in wired and wireless networks, it now becomes possible to perform array signal processing of audio using networked sensors, actuators and computers. This paper presents components of a prototype system developed to support hands-free audio capture for audio recording and voice recognition based on multiple wirelessly networked laptops with onboard microphone arrays. Several key technologies demonstrated in this system include: 1) time synchronization scheme for distributed audio input devices that uses wireless network; 2) localization algorithms to reconstruct the geometry of audio sensors and speakers in a room; 3) cascaded beamforming algorithms for signals captured by distributed microphone arrays. This paper discusses both theoretical and practical aspects of mapping array signal processing algorithms on a distributed network of general-purpose computers with integrated audio sensors. Our experimental results demonstrate great potential of distributed audio arrays for hands-free command-and-control when compared to an array located on a single platform.

1. INTRODUCTION

Arrays of audio/video sensors and actuators such as microphones, cameras, loudspeakers and displays along with array processing algorithms offer a rich set of new features for emerging applications. Traditionally, array processing required expensive dedicated multi-channel I/O cards as well as expensive high-throughput computing systems due to the requirement to process all channels on a single machine. Recent advances in mobile computing and communication technologies, however, suggest a novel and very attractive platform for implementing these algorithms. Students in classrooms and co-workers at meetings are nowadays accompanied by several mobile computing and communication devices with multichannel audio and video I/O capabilities onboard such as laptops, PDA's, and tablets. In addition, high-speed wireless network connections, like IEEE 802.11a/b/g, are available to network those devices. Such ad-hoc sensor/actuator networks can enable emerging applications that include multi-stream audio and video, smart audio/video conference rooms, meeting recordings, automatic lecture summarization, hands-free voice communication, speech enhancement and object localization. No dedicated infrastructure in terms of the sensors, actuators, multi-channel interface cards and computing power is required. In order to enable such attractive teleconference and multimedia applications, several important technical and theoretical problems are needed to be addressed before the idea of using those devices for array DSP algorithms can materialize in real-life applications. Among those, three of the most important problems are (1) to synchronize a network of distributed computers to provide a common reference time for the collaborative array processing, (2) to determine the 3D positions of all sensors and actuators in a room to provide a common space for multichannel audio processing and (3) to perform collaborative beamforming based on the distributed arrays to get high quality audio perception. In the following sections, we will describes our proposed solutions to these three problems. Simulations are also shown to demonstrate our ideas.

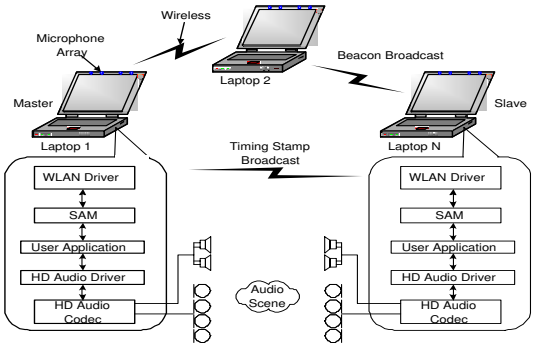

Fig. 1. Distributed audio rendering/capturing system setup

2. MOBILE PC PLATFORMS

Our proposed distributed audio capturing system is shown in Fig. 1, which consists of N laptops. Each mobile PC platform has integrated multichannel sensing, processing, and communication capabilities.It has a built-in microphone array for recording acoustic signals and the capacity to support 5.1/7.1 channel audio outputs, instead of the traditional mono microphone input and stereo audio output. The audio codec supports a sampling rate ranging from 8 kHz to 48 kHz with a sample format of 16$\sim$24bit integer. Each platform is equipped with the WLAN card for high speed data exchange across nodes. Currently, mobile PC CPUs and Chipsets can support up to 4GB memory and provide sufficient computational resource for digital signal processing. Thus, mobile PC becomes a powerful platform for distributed sensor processing because it is not constrained by energy, processing power, and bandwidth. Self-organizing a distributed system based on GPCs to enhance the audio perception and teleconference experience becomes possible. However, in order to achieve collaborative array processing

among distributed nodes, we need to overcome the following challenges: time synchronization of mobile PC platforms, 3D localization among inter-nodes, and distributed cascaded beamforming. The rest of this paper will discuss those problems in details.

3. TIME SYNCHRONIZATION FOR PC PLATFORMS

The problem of time synchronization in large distributed computing systems is solved in practice using NTP and GPS systems. The precision of NTP (Network Time Protocol) is in the millisecond range and is not sufficient for many audio array processing algorithms. Global Positioning system (GPS) provides a much higher clock resolution but only works reliably outdoors and thus does not completely fit our application scenarios. Moreover, most clock synchronization algorithms studied in literature only address the problem of providing a common clock to distributed computing platforms but did not answer how to distribute this clock to I/O devices on the platform. To synchronize distributed microphone arrays we employ our method proposed in [1] and provide main steps of the algorithm in this section.

Let t be a global time and let $\theta^{Ref}(t)$ be the value of reference clock at time t. We use $\tau_{i,j}(t)$ to denote the number of samples since the start of the I/O produced by j-th A/D (or consumed by D/A) converter on i-th platform at time t. The overall synchronization problem is to estimate $\hat{\theta}_{i,j}^{Ref}(\tau_{i,j})$ (the global time stamp of audio sample τ_i) so that the difference $|\hat{\theta}_{i,j}^{Ref}(t) - \theta^{Ref}(t)|$ is minimized. Our system tackles this problem in two steps: (1) (inter-platform) the local CPU clocks (e.g., RDTSC - Read Time-Stamp Counter) of computers are synchronized against a reference clock, and (2) (intra-platform) I/O is synchronized against the local clocks and thus also against the global clock. Note that two different timing models are required since the I/O devices on a typical PC platform have their own internal clocks that is not synchronized to other platform clocks such as the RDTSC.

Let $\theta_i(t)$ be the value of i-th GPC local clock at time t. Inter-platform and intra-platform synchronization require finding estimates $\hat{\theta}_i^{Ref}(\theta_i)$ (convert value of local clock to global time) and $\hat{\theta}_{i,j}(\tau_{i,j})$ (convert sample number to local time stamp), respectively. Estimation process is required to satisfy several important constraints, i.e., monotonicity and smoothness [1].

In practice we use piecewise-linear models both for inter- and intra platform synchronization clock models. Therefore we need to find the model parameters a_i and b_i such that

$$\hat{\theta}_i^{Ref}(\theta_i) = a_i(\theta_i)\theta_i + b_i(\theta_i), \quad (1)$$

and $\alpha_{i,j}$ and $\beta_{i,j}$ such that

$$\hat{\theta}_{i,j}(\tau_{i,j}) = \alpha_{i,j}(\tau_{i,j})\tau_{i,j} + \beta_{i,j}(\tau_{i,j}). \quad (2)$$

The dependency of the model parameters on time approximates instabilities in the clock frequency due to temperature variations and other factors. In practice, these instabilities are in the order of 10^{-5}. This time dependency also signifies the need for their periodic updates as described in the following. Let a_i, b_i be current model parameters and a_i', b_i' be updated parameters at time θ_i^0. At first, a range in local clock values $[\theta_i', \theta_i'']$, $\theta_i^0 < \theta_i' < \theta_i''$ is chosen for transition from the "old" model to the "new" model.

For the synchronization of CPU/platform clocks over a wireless network we propose to use a series of arrival times of multicast packets sent by a wireless access point (AP). In our current approach we implement a pairwise time synchronization with one node chosen as the master (say $\theta^{Ref} = \theta_0$). All other nodes (clients) are required to synchronize their clocks to the master (laptop 1 in Fig. 1). The inter-platform clock synchronization algorithm is implemented in Synchronization Agent Module (SAM) and consists of the four steps: 1)AP sends next beacon packet; 2)Master node records its local time of packet arrival and distributes it to all other nodes; 3)Client nodes record both their local times of arrival of beacon packets from AP, and the corresponding times received from the master; 4)Clients update local timing models based on the set of local timestamps and corresponding master timestamps.

In order to synchronize the audio clock to the CPU clock we use a similar approach. The ISR of the audio driver is modified to timestamp the samples in the OS buffer using the CPU clock to form a set of observation pairs $(\tilde{\theta}_i^j, \tau_{i,k}^j)$, where j now represents the index of an audio data packet and k is the index of A/D (D/A) on i-th GPC. Except for the fact that the τ^j are available without any noise (it is simply the number of samples processed!) we are back to the problem of determining the linear fit parameters for pairs of observations that we solved in the previous section using the LTS method. In addition d_{isr} component (that is responsible for most of variability in delay) is likely to have the same statistics as the similar parameter in the network packet receiving scenario.

Our measurements performed on actual PC platforms show that at any time the audio I/O offset between different computers can be kept below 50 μs that enables usage of *distributed PC platforms* for wide variety of array signal processing including distribute beamforming discussed in this paper.

4. 3D LOCALIZATION

Most of the multi-microphone/loudspeaker array processing algorithms require that the position of the microphones/loudspeakers be known. Current systems either place the microphones in known locations or manually calibrate them. Distributed microphone and loudspeaker arrays formed by mobile computing platforms require different approach. We developed a system in which the positions of the sensors on different devices are automatically self-localized using the actuators present in the system.

The problem of self-localization of a network of nodes involves two steps: ranging and multilateration. Ranging requires estimating the distance between two nodes in the network. Multilateration refers to using the estimated ranges to find the position of different nodes. The ranging technology can be either based on the Time-Of-Arrival (TOA) or the Received Signal Strength (RSS) of acoustic, ultrasound or radio frequency (RF) signals. The choice of a particular technology depends on the environment and the range for which the sensor network is designed. Localization using GPS is not suitable for our applications since GPS systems do not work indoors and are very expensive. Also RSS based on RF is very unpredictable and the RF TOA is very small to be used indoors. Ideally, we would like to use the existing sensors and actuators on mobilePCs to estimate their positions. We implemented ranging technology based on acoustic TOA and Maximum Likelihood (ML) estimation to get the positions [2].

To introduce ML estimation idea we assume we have M microphones and S sources. Let $\mathbf{m_i}$ for $i \in [1, M]$ and $\mathbf{s_j}$ for $j \in [1, S]$ be the three dimensional vectors representing the spatial coordinates of the i^{th} microphone and j^{th} source, respectively. Let ξ be the $(M+S)\times 3$ matrix where each row is formed by the spatial coordinates of each sensor, i.e., $\xi = [\mathbf{m_1}, .., \mathbf{m_M}, \mathbf{s_1}, .., \mathbf{s_S}]^T$.

We excite one of the S sources at a time and measure the TOA at each of the M microphones. Let τ_{ij} be the estimated TOA and t_{ij} the actual TOA for the i^{th} microphone due to the j^{th} source. The actual TOA t_{ij} for the i^{th} microphone due to the j^{th} source is given by

$$t_{ij} = \frac{\parallel \mathbf{m_i} - \mathbf{s_j} \parallel}{c} \tag{3}$$

where c the speed of sound in the acoustical medium. It is often reasonable to assume that the measured TOA, τ_{ij} is corrupted by zero-mean additive white Gaussian noise n_{ij} with known variance σ_{ij}^2, i.e, $\tau_{ij} = t_{ij} + n_{ij}$. The Maximum Likelihood estimate (ML) $\hat{\xi_{ML}}$ can be written as:

$$F_{ML}(\xi) = \sum_{j=1}^{S} \sum_{i=1}^{M} \frac{(\tau_{ij} - t_{ij})^2}{\sigma_{ij}^2} \tag{4}$$

$$\hat{\xi_{ML}} = \arg_{\xi} \min[F_{ML}(\xi)] \tag{5}$$

The ML estimate for the node coordinates of the microphones and loudspeakers is implicitly defined as the minimum of the nonlinear function given in Equation 5. This function has to be minimized using numerical optimization methods. Least squares problems can be solved using a general unconstrained minimization. However there exist specialized methods like the Gauss-Newton and the Levenberg-Marquardt method which are more efficient. The Levenberg-Marquardt method [3] is a popular method for nonlinear least squares problems. It is a compromise between steepest descent and Newton's methods. The steepest descent method potentially has a very slow convergence, but can converge from any starting point. Newton's method converges fast but requires a good initial guess and computation of the inverse of the Hessian matrix.

In real system settings we found our localization method capable of locating microphones within a few centimeter precision.

5. DISTRIBUTED CASCADED BEAMFORMING

After the common time and accurate position knowledge of sensors and desired source are obtained, beamforming is performed to improve the SNR and speech intelligibility. In general cases, audio signals captured by each microphone sensor on each GPC node are transmitted to the master node, then wide-band beamforming is applied to all the data to form the beamformer output,

$$Y(w) = \sum_{i=1}^{N} \sum_{j=1}^{L} W_{ij}(w) X_{ij}(w) \tag{6}$$

where $Y(w)$is the beamformer output in frequency domain, $X_{ij}(w)$ denotes the frequency domain audio signal captured by the jth microphone at the ith GPC node, $W_{ij}(w)$ is the beamforming weight chosen according to certain criterion, and L is the microphone number at each node. Here we assume that each node has the same microphone array setup. The drawback of this centralized processing mode is that all the processing loads occur at the single master mode and transmission of all the captured audio raw data to master node would require high network bandwidth.

To overcome that, through sufficiently utilizing GPC computing power, we propose the distributed cascaded beamforming scheme. It can be observed that the beamforming weight $W_{ij}(w)$ in Equation 6 can be separated into two parts,

$$W_{ij}(w) = A_i(w) H_{ij}(w) \tag{7}$$

where $H_{ij}(w)$ is the frequency response of the local beamforming on the microphone array of the ith node and $A_i(w)$ is the frequency response of the internode beamforming on the distributed node array. So the beamforming becomes a two-step cascaded processing: First, each node performs local beamforming to form a beam to the target source; Second, each node transmits its local beamforming output to the master node and the internode beamforming is then applied to get enhanced audio.

For the first step, due to the small-size microphone array used on GPC, the far-field assumption is valid and traditional robust adaptive beamforming techniques [4, 5] can be utilized to aggressively suppress interference and noise. For the second step, the internode beamforming should be a near-field beamforming problem because the interspacing between two nodes is usually several meters and comparable to the distance between source and GPC node. As local beamforming has good noise reduction performance, the simple *near-field compensation* method is sufficient to compensate for propagation delays to further improve the SNR. The beamforming weight $A_i(w)$ is only related to the distributed system geometry and source position,

$$A_i(w) = B_i exp(jwr_{si}/c) \tag{8}$$

where r_{si} denotes the distance between target source and the ith node, c is the sound speed and B_i is the scale weight of the ith node which is related to r_{si}. Assuming the source is in the free field with spatially uniformly distributed noise whose variance is σ^2, due to the different distances to the source, each node has different input SNRs which will result different array gain and output SNR after local beamforming. After compensating the propagation delays, the internode beamforming scale weights are selected by maximizing total output SNR, equivalent to

$$\min[\sum_{i=1}^{N} B_i^2 \sigma_{out}^{(i)2}] \quad with \quad \sum \frac{B_i}{r_{si}} = 1 \tag{9}$$

where $\sigma_{out}^{(i)2}$ is the noise variance at local beamforming output. By assuming equal output noise variance at all nodes, we have

$$B_i = \frac{1}{r_{si}} (\sum \frac{1}{r_{si}^2})^{-1}. \tag{10}$$

We get the conclusion that the node with shorter distance to source will have larger weight and make more contribution to the internode beamforming, which is very explicit because that node will have higher SNR output. In other words, this means that the distributed system enlarges the operating range comparing to the single microphone array system and even GPC far from the source can experience the high SNR audio that is only for the single microphone array system close to the source. And the distributed system can also obtain the extra gain ($\sim$N) from the internode beamforming.

By applying distributed cascaded beamforming, the computing burden is evenly distributed to each node and the processing power of each GPC can efficiently exploited. The bandwidth requirement is reduced to $1/L$ times of that of the centralized processing mode because only one channel audio data is needed to be transported from each node to the master node. Moreover, distributed cascaded beamforming can prevent grating lobes resulted by spatial alias. As we know, distributed system usually has large interspacing among nodes which will result in spatial under-sampling comparing to the sound wavelength. However, the

linear microphone array on each node is of small size and thus ambiguity-free. So no spatial alias occurs by combining linear microphone array with sensor network. Another important advantage of our design is that our system is robust to the 3D localization errors. The reason is that the adaptive beamforming algorithm [4, 5] used in linear array can track the target source's direction when there is small mismatch between assuming direction and actual direction.

6. EXPERIMENTAL RESULTS

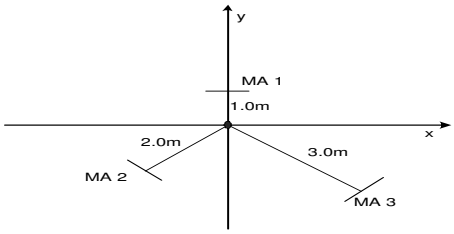

Fig. 2. Example of distributed array and target source configuration.

Fig. 2 shows an example of a distributed microphone array system. There are three GPC nodes with distances 1m, 2m and 3m to the origin of the coordinate, respectively. The angles between GPC nodes and axis x are 90 degree, 210 degree, and 330 degree. On each GPC node, a 4-sensor microphone array is equipped with the inter-element spacings 4.5cm, 7.0cm and 4.5cm. The target source is assumed to lay on the origin in the free-field. Diffuse Gaussian white noise is added to every microphone to simulate the ambient noise. Due to the distance differences from source to GPC nodes, the input SNRs of the three nodes are 7dB, 1dB, and -3dB respectively.

In the first experiment, we assume there is no time synchronization or localization errors. We perform local GSC beamforming [5] and distributed cascaded beamforming to investigate the benefit of distributed processing comparing with the single microphone array processing. Fig. 3 depicts the output SNRs of local beamforming at each GPC node and distributed cascaded beamforming on the whole system. It can been seen that distributed microphone array is more beneficial to the situation that the source is far from the single microphone array, i.e. the input SNR is low. The reason is that distributed microphone array can leverage the local beamforming contribution of the node closer to the source. Thus, in another word, distributed microphone array expands the operation range of traditional single array. In the second experiment, we study the sensitivity of distributed cascaded beamforming to node localization errors. We approximated distance and direction estimation errors of the 3D localization with Gaussian distribution with the standard deviation of 0.1m (in practice numbers are usually smaller). We investigate the distributed cascaded beamforming performance when the direction estimation standard deviation varies from 0 degree to 20 degree. The result of Output SNR versus direction estimation standard deviation is shown in Fig.4 4. We see that the proposed distributed cascaded beamforming is robust to the direction estimation deviation. It is mainly due to the robustness and tracking ability of local GSC beamforming under direction mismatch.

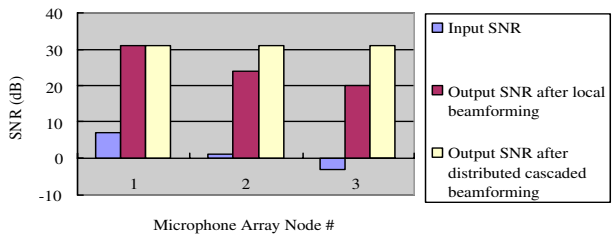

Fig. 3. SNRs for local and distributed cascaded beamforming.

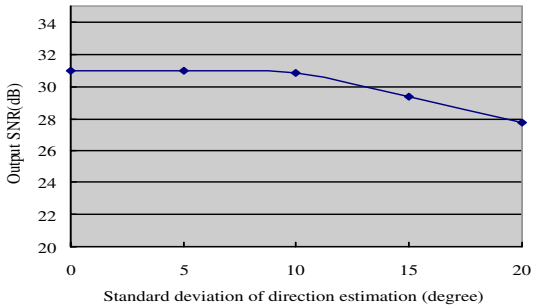

Fig. 4. Sensitivity of distributed cascaded beamforming to direction estimation deviation

7. SUMMARY

In this paper we described several important components (time synchronization, 3D localization and distributed beamforming algorithm) that enable array processing on distributed network of sensors and actuators formed by mobile computing platforms. Our results indicate a great potential of such systems for future multimedia applications in home and office environments.

8. REFERENCES

[1] D. Budnikov et al., "Providing common I/O clock for wireless distributed platforms," in *Proc IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, 2004, vol. 3, pp. 909–912.

[2] V. Raykar, I. Kozintsev, and R. Lienhart, "Position calibration of audio sensors and actuators in a distributed computing platform," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 70–83, Jan 2005.

[3] D. P. Betrsekas, *Nonlinear Programming*, Athena Scientific, 1995.

[4] O. Hoshuyama, A. Sugiyama, and A. Hirano, "A robust adaptive beamformer for microphone arrays with a blocking matrix using constrained adaptive filters," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2677–2684, Oct 1999.

[5] Z.R. Hou, Y. Jia, and M. Yeung, "Frequency-domain gsc beamforming with enhanced robustness against mistracking," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, (IWASPAA 2003)*, Oct 2003, pp. 33–36.