Voice Extraction by On-Line Signal Separation and Recovery

G. Erten, *Senior Member, IEEE*, and F. M. Salam, *Fellow, IEEE*

Abstract—The paper presents a formulation and an implementation of a system for voice output extraction (VOX) in real-time and near-real-time *realistic* real-world applications. A key component includes voice-signal separation and recovery from a mixture in practical environments. The signal separation and extraction component includes several algorithmic modules with a variety of sophistication levels, which include dynamic processing neural networks in tandem with (dynamic) adaptive methods. These adaptive methods make use of optimization theory subject to the dynamic network constraints to enable practical algorithms. The underlying technology platforms used in the compiled VOX software can significantly facilitate the embedding of speech recognition into many environments. Two demonstrations are described: one is PC-based and is near-real-time, the second is digital signal processing based and is real time. Sample results are described to quantify the performance of the overall systems.

Index Terms—Adaptive networks, audio signal processing, DSP, gradient descent, independent component analysis, neural networks, nonlinear networks and systems, optimization, speech processing, state-space models, statistical independence criteria.

I. INTRODUCTION

THE ABILITY to selectively enhance audio signals of interest while suppressing spurious ones is an essential prerequisite to widespread practical use of far-field voice-activated systems. Such audio signal discrimination allows for selective amplification of a single source of speech within a mixture of two or more signals, including noise and other speakers' voices. Although speech recognition systems have made significant progress in the last few years, their sensitive dependence on the quality of the voice signal still prevents them from being widely deployed in man-machine communication. In addition, the success rate for recognition, especially in the case of large vocabulary and/or continuous speech recognition, is simply unsatisfactory to the end user in many environments. Due to the need of having pure voice signals, the users of these systems need to wear headsets with microphone attachments. This is often quite restrictive and unnatural. Thus, freedom from headsets is one significant and driving concern.

For removal of noise from the audio signal transduced through a microphone, most traditional signal processing systems use (linear) frequency and filter-based techniques. This

approach has obvious limitations, especially when the spectral content of the voice overlaps with other sounds (including those produced by other speakers) in the background. By contrast, the methods introduced here are directly nonlinear time-domain based, which extract and track voice signals of interest based on alternate signal. One of the primary components of our voice-signal enhancement module involves the extraction of a speech signal from transduced sound mixtures. Techniques of this nature have widely been called independent signal separation and recovery in the literature [1]–[4].

The signal-extraction component may be intuitively described as follows: several unknown but independent temporal signals propagate through a mixing and/or filtering, natural or synthetic medium.

By sensing the outputs of this medium, a network (e.g., a neural network, a system, or a device) is configured to counteract the effect of the medium and adaptively recover the original unmixed signals. The property of signal independence, with the possibility of minimizing signal dependency, is assumed for this processing. No additional *a priori* knowledge of the original signals is assumed. This processing represents a form of self (or unsupervised) learning. The weak assumptions and self-learning capability render such a network attractive from the viewpoint of real-world applications where (off-line) training may not be practical.

The blind-separation approach has great advantages over the existing adaptive filtering algorithms. For example, when the mixture of other signals is labeled as noise in this approach, no specific *a priori* knowledge about any of the signals is assumed; only that the processed signals are independent. This is in contrast to the noise-cancellation method proposed by Widrow *et al.* [5], which requires that a reference signal be correlated exclusively to the part of the waveform (i.e., noise) that needs to be filtered out. This latter requirement entails specific *a priori* knowledge about the noise, as well as the signal(s).

The separation of independent sources is valuable in numerous and major applications in areas as diverse as telecommunication systems, sonar and radar systems, audio and acoustics, image/information processing, and biomedical engineering. Consider, e.g., the scenario of audio and sonar signals, where the original signals are sounds, and the mixed signals are the output of several microphones or sensors placed at different vantage points. A network will receive, via each microphone, a mixture of sounds that are usually delayed relative to one another and/or relative to the original sounds. The network's

Manuscript received November 1, 1998; revised March 9, 1999. This paper was recommended by Guest Editors F. Maloberti and R. Newcomb.

G. Erten is with IC Tech, Inc., Okemos, MI 48864 USA.

F. M. Salam is with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA.

Publisher Item Identifier S 1057-7130(99)05641-4.

role is then to dynamically reproduce the original signals, where each separated signal can be subsequently channeled for further processing or transmission. Similar application scenarios can be described in situations involving heart-rate measurements, communication in noisy environments, engine diagnostics, and uncorrupted cellular phone communications.

The paper is organized as follows: Section II provides a brief description of the problem of speech signal separation in practical hands-free environments. Section III describes modeling of both the environment and, consequently, the processing network using the state-space approach. Feedforward and feedback structures are considered, as well as their generalization to nonlinear parameterized models. Section IV describes the framework and derivation of the update laws, expressed as optimization of a performance index subject to the dynamics of the processing network. This encompasses the most general and advanced update laws. In fact, families of update laws are analogously generated for a variety of mixing environments. Section V describes two example demonstrations, one focusing on a digital signal processing (DSP) system, and the other on a standard PC platform, both interfaced with microphones and speakers. These demos serve as testbeds for the modular codes and methodology which can be integrated as part of a system for voice extraction and tracking. In Section VI, we summarize our concluding remarks.

II. SEPARATION AND EXTRACTION OF SPEECH SIGNALS

The recovery and separation of independent signal sources is a classic but difficult problem in signal processing. The problem is complicated by the fact that in practical situations, many relevant characteristics of both the signal sources and the mixing medium are unknown. Two main categories of methods exist in the literature:

- 1) conventional discrete signal processing;
- 2) neurally inspired adaptive algorithms.

Conventional signal-processing approaches [1] to signal separation originate in the discrete domain in the spirit of traditional digital signal-processing methods that use statistical properties of signals. Such signal-separation methods employ discrete signal transforms and *ad hoc* filter/transform function inversion. Statistical properties of the signals in the form of a set of cumulants are used and these cross cumulants are mathematically forced to approach zero. This constitutes the crux of the family of algorithms that search for the parameters of, mostly, finite-impulse response (FIR), transfer functions that recover and separate the signals from one another. Calculating all possible cross cumulants, on the other hand, would be impractical and too time consuming for real-time implementation. In addition, the methods discussed in [1] do not provide an extension beyond the two mixture case, i.e., three or more signals mixed together cannot be separated in this fashion. However, a possible extension can be developed, but ends up being computationally too expensive. Neurally inspired adaptive algorithms pursue an approach originally proposed by Herault and Jutten, now called the Herault-Jutten (HJ) algorithm [2], [3]. However, the HJ algorithm

has been considered heuristic with suggested adaptation laws that have been shown to work mainly in special circumstances. The theory and analysis of prior work pertaining to the HJ algorithm are still not sufficient to support or guarantee the success encountered in experimental simulations. Their proposed algorithm assumes a linear static medium with no filtering or delays. Specifically, the original signals are assumed to be transferred by the medium via a matrix of unknown but constant coefficients. To summarize, the HJ method: 1) is restricted to the full rank and linear static mixing environments; 2) requires matrix-inversion operations; and 3) does not take into account the presence of signal delays. Many approaches in the current literature pursue a modeling of the mixing medium in the form of an unknown constant matrix. In many practical applications, however, delays do occur, and on many occasions, the medium mixing environment may exhibit nonlinear phenomena. Accordingly, the previous work fails to successfully separate signals in many practical situations and real-world applications.

III. DYNAMIC MODELS FOR THE ENVIRONMENT AND THE PROCESSING NETWORK

The static mixing case studied by many, e.g., [2], [3], [6], is limited to mixing by a constant matrix. If we define the source signal vector $s(t)$ as $[s_1(t), \dots, s_N(t)]^T$, then the mixed-signal vector $M(t)$ is defined as

$$M(t) = \overline{D}s(t). \quad (1)$$

Separation of statically mixed signals is of limited use because of additional factors involved in superposition of signals in real mixing environments. Some examples of additional factors to be considered include the: 1) propagation time delays between sources and receivers or sensors; 2) nonlinear nature of the mixing functions introduced by the mixing medium as well as the signal sensors or receivers; and 3) unknown number of source signals that are to be separated. The methodology developed by our team addresses all three by first extending the formulation of the problem to include dynamic modeling of the signal mixing/interference medium. The dynamic portion of the mixing model we present in this paper accounts for more realistic mixing environments, defines their dynamic models, and develops an update law to recover the original signals within a comprehensive framework. In the dynamic case, the mixing environment is no longer a constant matrix. In fact, the dynamic representation of the signal mixing and separation processes takes the problem out of the realm of algebraic equations to the realm of differential (or difference) equations. Several state-space formulations have been reported in [7]–[9] and the references therein.

A. A Simple Dynamic Realization

We now discuss one simple complete formulation as an example. Recall that a feedback separation structure for the static case of (1) is given as (see [2])

$$U(t) = -DU(t) + M(t) \quad (2i)$$

which may be rewritten as

$$U(t) = [I + D]^{-1}M(t) = [I + D]^{-1}\overline{D}s(t). \quad (2ii)$$

This yields output vector $U(t)$, which estimates the original signal sources $s(t)$ by adaptively updating the entries of a matrix D in (2ii), so that

$$[I + D]^{-1}\overline{D} = P_1 \quad \text{or} \quad D = \overline{D}P_2 - I \quad (2iii)$$

where P_1 and P_2 are permutation matrices of a (nonsingular) diagonal matrix. A special case of P_1 and P_2 would be permutation matrices of the identity.

As introduced in [4], we view (2i) as a limit of the dynamic equation

$$\tau \dot{U}(t) = -U(t) - DU(t) + M(t) \quad (3)$$

where τ is a small time constant. This facilitates the computation by initializing the differential equation in (3) from an arbitrary guess. It is important, however, to ensure the separation of time scales between (3) and the adopted update procedure of D like the one defined below by (4). This may be ensured by making in (4) sufficiently small. Note that d_{ij} is the ij th component of the matrix D .)

$$\dot{d}_{ij} = \eta f_i(u_i(t)) g_j(u_j(t)) \quad (4)$$

where η is sufficiently small, and $f_i(\cdot)$ and $g_j(\cdot)$ are a pair of a family of odd functions [2], [3]. In particular, we use a family of functions we developed in [7], which includes

$$f_i(u_i) = \begin{cases} \sinh(u_i), \tanh^{-1}(u_i), \text{their inverses, or} \\ \sum_{i=0}^N \frac{u^{(2i+1)}}{(2i+1)!} \end{cases} \quad (5)$$

$$g_j(u_j) = \begin{cases} \sinh^{-1}(u_j), \tanh(u_j), \text{their inverses, or} \\ u_j \end{cases} \quad (6)$$

When using (4) for the static case, one solves for $U(t)$ from (2ii). For the dynamic case, however, (3) is used instead.

The procedure thus enumerates the N differential equations of (3). In addition, the adaptation process for the entries of the matrix D can be defined by multiple criteria, e.g., the selection of functions $f_i(\cdot)$ and $g_j(\cdot)$ in (4). The process facilitates the computation by initializing the differential equations from an arbitrary guess, and makes it possible to construct continuously adaptive algorithms [4], [7]. Many types of approaches to solving such differential equations exist. One can distinguish methods as continuous versus discrete as well as fixed versus variable step sizes.

B. The State-Space Approach

Let the n -dimensional source signal vector be $s(t)$, and the m -dimensional measurement vector be $M(t)$. Let the mixing environment be described by the linear time-invariant (LTI) state-space [4], [8]

$$\dot{\overline{X}} = \overline{A}\overline{X} + \overline{B}s \quad (7i)$$

$$m = \overline{C}\overline{X} + \overline{D}s. \quad (7ii)$$

(Note that we have suppressed the dependence on time of the variables for simplicity of presentation.) Assume that the

state is of dimension $N > n$. The parameter matrices $\overline{A}, \overline{B}, \overline{C}$, and $\overline{D}$ are of compatible dimensions [8]. This formulation encompasses both continuous-time and discrete-time dynamics. The dot on the state $\overline{X}$ means derivative for continuous-time dynamics, it however means “advance” for discrete-time dynamics. The mixing environment is assumed to be (asymptotically) stable, e.g., the matrix $\overline{A}$ has its eigenvalues in the left half complex plane in the continuous-time case, and analogously within the unit (complex) circle in the discrete-time case. We now consider two processing network structures.

1) *The Feedforward Network Structure/Architecture:* The (adaptive) feedforward network is proposed to be of the state-space form

$$\dot{x} = Ax + Bm \quad (8i)$$

$$y = PCx + Dm \quad (8ii)$$

where y is the n -dimensional output, x is the internal state, and the parameter matrices are of compatible dimensions [8]. For simplicity, let us assume that x has the same dimensions as $\overline{X}$. Several adaptive laws have been reported in [7], [9], [10] and the references therein.

2) *The Feedback Network Structure/Architecture:* The (adaptive) feedback structure is defined to be of the form

$$\dot{x} = Ax + By \quad (9i)$$

$$y = m - Cx - Dy \quad (9ii)$$

where y is the n -dimensional output, x is the internal network state vector which is of dimension higher than or equal to N , and the parameter matrices (A, B, C , and D) are of compatible dimensions. For simplicity, we assume that X has the same dimensions as $\overline{X}$. Several adaptive laws have been reported in our previous work in [7], [9], [10] and the references therein.

In a more general network framework, the dynamic network can be represented by the nonlinear time-varying state representation with parameters as

$$\dot{x} = f(t, x, y, w_1) \quad (10i)$$

$$y = g(t, x, m, y, w_2) \quad (10ii)$$

where $f(\cdot)$ and $g(\cdot)$ are differentiable functions which permit existence and uniqueness of solutions of the system of equations. Such a nonlinear network may be used to counteract the nonlinearity and dynamics of the environment which may be a generalization of the LTI system in (7i) and (7ii). We note also that w_1 and w_2 represent the parameters in the state and output equations, respectively.

IV. FORMULATION OF THE UPDATE LAWS

For the dynamic environment and processing networks, the original update laws used for the static case or the simple dynamic case can not be expected to work for general cases. The appropriate formulation is to consider an optimization process of the dependence criterion under dynamic network constraints.

The mutual information of a random vector y is a measure of *dependence* among its components and is defined as ([6],

[7], [9], [10], and the references therein)

$$I(\mathbf{y}) = \int f_y(\mathbf{u}) \ln \frac{f_y(\mathbf{u})}{\prod_i f_{y_i}(u_i)} d\mathbf{u} \quad (11)$$

where $f_y(\mathbf{y})$ is the probability density function (pdf) of the random vector $\mathbf{y}$. The functional $I(\mathbf{y})$ is always nonnegative and is zero if and only if the components of the random vector $\mathbf{y}$ are statistically independent. This important measure defines the *degree of dependence* among the components of the signal vector. Therefore, it represents an appropriate functional for characterizing (the degree of) statistical independence. $I(\mathbf{y})$ can be expressed in terms of the entropy

$$\varphi(y) \propto I(y) = -H(y) + \sum_i H(y_i) \quad (12)$$

where $H(\mathbf{y}) := -E[\ln f_y]$ is the **entropy** of $\mathbf{y}$, $H(y_i)$ is the marginal entropy of the component signal y_i , and $E[\cdot]$ denotes the expected value. We also define the measure of dependence $\varphi(y)$ to be proportional to $I(y)$.

The update law is now developed for dynamic environments to recover the original signals following the procedures in [7], [9], [10]. Let the network be a continuous-time dynamical system. One defines the performance index as in optimization theory [11] as

$$J(\mathbf{x}, \mathbf{w}) = \int_0^T L(t, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{y}, \lambda, \mathbf{w}) dt \quad (13)$$

where L is the Lagrangian and is defined as

$$L(t, x, \dot{x}, y, \lambda, w) = \varphi(y, w_2) + \lambda^T (f(t, x, y, w_1) - \dot{x}) \quad (14)$$

where $\lambda(t)$ is the “adjoint” state [11]. In general, the dynamics are described by the Euler–Lagrange variational equations

$$\frac{\partial L}{\partial \lambda} - \frac{d}{dt} \left\{ \frac{\partial L}{\partial \dot{\lambda}} \right\} = 0 \Rightarrow \dot{x} = f(t, x, y, w_1) \quad (15i)$$

$$\frac{\partial L}{\partial x} - \frac{d}{dt} \left\{ \frac{\partial L}{\partial \dot{x}} \right\} = 0 \Rightarrow -\dot{\lambda} = \frac{\partial f^T}{\partial x} \lambda + \frac{\partial \varphi}{\partial x} \quad (15ii)$$

with boundary conditions $x(0) = 0$ and $\lambda(T) = 0$. The parameter update of $w_p, p = 1, 2$, according to the general gradient (instantaneous) descent form, is

$$\dot{\mathbf{w}}p = -\eta \frac{\partial L}{\partial \mathbf{w}p}. \quad (16i)$$

A variant of (16i) used in practice includes a sufficiently small leakage (i.e., damping) term as

$$\dot{\mathbf{w}}p = -\eta \frac{\partial L}{\partial \mathbf{w}p} - \alpha \mathbf{w}p. \quad (16ii)$$

As an example, consider the specific case when the environment is considered a linear dynamical system. The network, consequently, may be modeled as a linear (feedforward) dynamical system.

The adjoint state equation in that case is given by

$$-\dot{\lambda} = A^T \lambda + \frac{\partial \phi}{\partial \mathbf{x}}. \quad (17)$$

The functional ϕ represents a scaled version of our measure of dependence $I(y)$ and $\mathbf{w}$ is a vector constructed of the rows of the parameter matrices C and D . Note that a canonical realization [8] may be used so that B is constant. The matrix A

in the canonical representation may have only N -parameters, where N is the dimension of the state vector x . The parameters A, C , and D , represented generically by $w_p, p = 1, 2$, will be updated using the general gradient descent form.

Consequently, using the performance index defined in (12), the matrices C and D are thus updated according to [12]

$$\dot{D} = \eta (I - f_a(\mathbf{y}) \mathbf{y}^T) D \quad (18)$$

$$\dot{C} = \gamma (I - f_a(\mathbf{y}) \mathbf{x}^T) C \quad (19)$$

where I may be represented by a scaled version of the identity matrix, η and γ are update rate parameters or functions, and $f_a(\cdot)$ is given by a variety of nonlinear expansive or compressive odd-functions, which include hyperbolic sine and tangent and their inverses. Or, in general, sigmoidal or inverse of a sigmoidal functions. In the specific computation/approximation performed in [7], [9], [10], one function used is given as

$$f_a(y) = \frac{71}{12} y^{15} - \frac{355}{12} y^{13} + \frac{190}{3} y^{11} - \frac{4033}{24} y^9 + \frac{941}{3} y^7 + \frac{47}{8} y^5 + y^3 + y. \quad (20)$$

The essential features in using (20) are summarized as follows:

- 1) it is analytically derived and justified;
- 2) it includes a linear term in y and thus enables the performance of second-order statistics necessary for signal whitening;
- 3) it contains higher order terms which emanate from the fourth-order cumulant statistics in the output signal y ;
- 4) it does not make the assumption that the output signal has unity covariance.

To our knowledge, the function of (20) represents the only analytically derived function in the literature with the above characteristics, to date. This function, therefore, avoids the limitations of another analytically derived function reported in [6]. Computer simulations confirm that the algorithm converges if the function defined in (20) is used. Examples of computer simulations were reported in [7], [9], [10] and the references therein.

V. DEMONSTRATION ENVIRONMENTS

In this work, the voice-extraction system is demonstrated under both simulated and real dynamic multimicrophone mixing conditions in two environments: 1) a near-real-time PC environment and 2) a real-time DSP environment. In the demonstration, a tradeoff between efficiency of computation and acceptable performance is judiciously used to render a real-world operational system.

A. Static and Real Dynamic Mixing Conditions

Simulated mixing involves an audio-file mixing program for the PC environment and an audio mixer for the DSP environment. In both cases, the effect of static mixing of sources is a point-by-point addition, such that each mixture contains a differently weighted combination of the two original sound sources. There is a singularity constraint imposed on the ratios for signal separation to succeed. One model-mixing

Fig. 1. Dynamic real mixing setup. (a) Two types of four microphone arrangements and respective amplifiers are shown. These microphones sense the mixed signals from two speech sources from the speakers. Either the one- or two-dimensional arrangement can be used. (b) The speaker setup used in the PC test environment is illustrated.

matrix (with relevance to practical considerations) is

$$\begin{bmatrix} 1.0 & 0.9 \\ 0.8 & 0.9 \end{bmatrix}$$

Thus, the resulting inputs to signal separation can be represented as follows:

$$\text{Mixture 1} = 1.0 \text{ source1} + 0.9 \text{ source2}$$

$$\text{Mixture 2} = 0.8 \text{ source1} + 0.9 \text{ source2}$$

Real dynamic multimicrophone mixing, on the other hand, involves two types of four microphone arrangements, as illustrated in Fig. 1(a) and (b). All microphones used are of the inexpensive condenser variety, which usually cost less than \$1.00 each. The algorithms used by the codes do not require high-fidelity audio input. The voice-extraction program first selects two microphone outputs from the available four. This selection process is used to minimize the computation time.

B. The PC Environment

The PC environment consists of:

- 1) an A/D data-acquisition card to collect the data from the arrangement;
- 2) a PC host for data collection and processing.

The voice-extraction program runs on the central processing unit (CPU) of the PC host which, in this case, is a 300-MHz Intel Pentium II processor with MMX. The four microphone outputs comprise the only input data to the voice-extraction program. The four sound segments are stored in audio files of .wav format. The data sampling rate is 22 kHz and the resolution is 8 or 16 bits/sample.

C. Real-Time DSP Environment

The real-time DSP environment illustrated in Fig. 2 consists of:

Fig. 2. Schematic depiction of the real-time DSP environment.

- 1) a real-time C32 DSP board containing a Texas Instruments C32 floating-point digital-signal processor, as well as dual A/D and D/A channels;
- 2) a PC, which loads the program onto the DSP card.

The DSP environment was assembled to demonstrate the algorithms with real audio signals on-line and in real time. This is a significant step toward practical implementations of this family of signal separation algorithms, since real-time on-line implementation is not only desired, but necessary in many applications. Moreover, given that DSP's are the standard execution platform in many types of contemporary signal processing, it is more likely that the algorithms will be embedded in a DSP-based system. The signal-separation algorithms are coded to be interpreted by the DSP. This code is loaded into program memory on board the development board via the PC interface. The DSP is then started to execute its codes using signals digitized by the A/D channels as inputs.

Fig. 3. Due to the proximity of the microphones, the two input mixtures are very similar. Their difference shows this clearly.

Each sampled input generates an interrupt for the DSP. Since this interrupt occurs every $20\ \mu\text{s}$ for the 50-kHz sampling rate used, the DSP has only $20\ \mu\text{s}$ to process each sample. For a DSP rated at 25 million instructions per second (MIPS), at most 500 instructions can be performed per sample point. Once programmed, the DSP operates independently from its PC host and receives inputs directly from the external environment, in this case, either from the multiple microphone amplifiers or from the audio mixer. The outputs produced are transferred immediately to the D/A output channels and can be heard through a speaker as they are produced.

VI. VOICE EXTRACTION RESULTS

A. The Voice-Extraction Program Execution: The voice-extraction program has four components, each dedicated to a specific function. These are: 1) signal selection; 2) pre-processing; 3) signal extraction; and 4) signal tracking. We focus here on the signal-extraction component.

As stated earlier, if the number of mixtures exceed two, the voice-extraction program first selects two microphone outputs from the four available. This selection process is needed to keep the computation time to a minimum. Once the voice-extraction program determines which two signals are to be used, some preprocessing is performed to reduce the inherent echo and remove high-frequency elements. The voice-signal extractor then uses dynamic signal-separation methods, described in part in Section III, to extract the individual speech segments from the mixtures. Once the voice-extraction program locks onto an individual speaker's voice signal, the signal is tracked by means of various signal tracking methods in combination with the dynamic signal separation technique.

No prior training procedures or explicit acoustic models are used. The extraction of the program is on-line and in real-time within the DSP environment. In the PC environment, execution of the program takes approximately as long as the duration of the speech signal itself. The short processing time represents considerable evidence that the processing can, in fact, be embedded as a front end to speech-recognition software for real-time and/or near-real-time processing. Following the PC host execution of the voice-extraction program, the resulting distilled speech signals can be played back through the PC host's sound card.

B. Multimicrophone Mixing Results in the PC Environment

From the PC environment, we present results of processing of 10-s segments of mixed speech recorded using the speaker positioning shown in Fig. 2(b). Two speech segments were played back through the speakers. One is that of a male speaker and the other is of a female speaker.

The three figures numbered Figs. 3–5 illustrate the results. Fig. 3 is included to illustrate that the two mixtures acquired from the selected microphones are actually very similar. Their difference shows that the individual speech signals are present in each sound mixture to approximately the same degree. Only a slight degree of dissimilarity is sufficient for the program to discriminate one speech segment against the other. Fig. 4 shows a 1 s signal segment, where only one of the speakers is active. The voice-extraction program is successful in selectively amplifying the signal in only one of the output channels, namely output channel 2. Output channel 1, where the other speaker's voice is convened, is nearly silent. In Fig. 5, one observes another signal segment where both speakers are active. One can observe that different segments are apportioned to the two channels. Audio playback confirms that each channel outputs primarily only one speaker's voice.

C. Real-Time Static-Mixing Results in the DSP Environment

Our current on-line demonstration has been taped and contains three distinct scenarios of static audio mixing and separation. First, a speech segment and a music recording are separated from one another. The mixer outputs contain combinations of these two sources on both the left and right channels. Algorithms running in the DSP environment separate these two, sending each separated audio signal to a single channel. One can clearly hear that the separation and recovery is successful. Second, a weak speech signal is recovered from a loud music segment background. The sound sources remain the same as those in the first scenario, yet the volume of the sound source supplying the speech input is reduced and the volume of the source supplying the music input is increased. This is carried out to the extent that the speech segment is no longer audible when one listens to either mixture. Then, algorithms executed on the DSP development environment recover the speech signal. When one listens to the output channel containing the speech segment, it is clearly audible and

Fig. 4. In this 1-s portion of speech, only one speaker is talking. The corresponding speech segment is routed to output channel 2. Output channel 1 is nearly silent.

Fig. 5. The two input mixtures are compared to the two output channels for approximately 1.25 s. One can observe visually that the portions of the mixtures are routed to different channels.

understandable. Third, the music input is replaced by another speech input. This time, the two sound sources play back tapes of the same speaker recorded saying different sentences. The voice-extraction algorithms running on the DSP development platform are successful in separating the two from each other. This is especially challenging, since the frequency profiles and other conventional signal characteristics are virtually the same for the two original segments and mixtures.

The described separation of signals is not possible with conventional signal-processing methods, e.g., frequency filters, Fourier analysis, etc., due to the significant overlap of frequency spectra, as well as nonideal mixing properties. This is illustrated conclusively in the third scenario of the demonstration, where two speech segments uttered by the same person are separated from each another.

In all three of these scenarios, we have utilized the same family of signal-separation algorithms. These algorithms are expandable to beyond two inputs, as long as multiple mixtures can be obtained. As a general rule of thumb in this implementation, the number of separated signals is less than or equal to the number of mixtures available. It is for this reason that we are proposing microphone arrays as a front-end to our process.

Fig. 6. A segment of speech mixed with multidirectional mechanical noise is extracted. The improvement in the SNR exceeds 17 dB.

D. Real-Time Multimicrophone Mixing Results in the DSP Environment

One of our latest multimicrophone mixing scenarios involve multiple noise sources and one speech signal about 1 in from the primary microphone of a four-microphone arrangement, similar to the one used for the PC environment. Four speakers placed in a ring around the microphone arrangement play back the same noise signal: a pre-recorded drill noise. The signal-to-noise ratio (SNR) level is slightly below 0 dB.

Figs. 6 and 7 illustrate the experimental results. In Fig. 6, a segment of speech is shown before and after voice extraction.

Fig. 7. A segment of audio where only the multidirectional mechanical noise is heard. The extracted voice output is silent, as expected. As before, the improvement in the SNR exceeds 17 dB.

The estimated improvement in SNR for this experiment is estimated to be over 17 dB. In Fig. 7, an audio segment composed entirely of noise (with the speech source silent) is shown. The voice-extraction output is nearly silent. It may be important to note that the adaptation of parameters continues during the silence period.

VII. CONCLUSION

We have presented a family of algorithms for dynamic environments and their derivations from the principles of optimization theory under the constraints of network dynamics. These derivations use general state-space models which encompass a variety of models. These models are more general than those reported in the literature.

In order to render the theoretical formulations meaningful in practice, we have steered the development toward realistic scenarios and computing platforms of both PC's and DSP's. Moreover, the algorithms developed have been incorporated as components of a modular code, which provides an engineering solution to voice extraction and a front end to speech recognition software in hands-free far-field environments.

To further improve the voice-extraction capabilities of the program, we are experimenting with alternate means and techniques for all four components of the program, i.e., microphone selector, voice-signal preprocessing, signal extraction, and voice tracking, to improve the output voice-signal quality. This paper has focused primarily on the voice-signal extractor component. We are also experimenting with more robust adaptation mechanisms for difficult and noisy contexts of the mixing environment. Our preliminary work indicated that one can effectively leverage additional computing steps for fault tolerance to significantly improve the robustness of the voice-extraction module. Moreover, the program can benefit from stochastic parameter update techniques, especially when based on multiple complementary criteria.

REFERENCES

- [1] E. Weinstein, M. Feder, and A. V. Oppenheim, "Multi-channel signal separation by decorrelation," *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 405–413, Oct. 1993.
- [2] C. Jutten and J. Herault, "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1–10, July 1991.
- [3] —, "Blind separation of sources, Part II: Problems statement," *Signal Processing*, vol. 24, no. 1, pp. 11–20, July 1991.
- [4] F. M. Salam, "An adaptive network for blind separation of independent signals," in *Proc. 1993 IEEE Int. Symp. Circuits and Systems*, vol. 1, pp. 431–434.
- [5] B. Widrow *et al.*, "Adaptive noise cancellation: principles and applications," *Proc. IEEE*, vol. 63, pp. 1691–1717, Apr. 1975.
- [6] S. Amari, A. Cichocki, and H. Yang, "A New Learning Algorithms for Blind Signal Separation," Cambridge, MA: MIT Press, 1996.
- [7] G. Erten and F. Salam, "Real time separation of audio signals using digital signal processors," presented at the IEEE MWSCAS '97, Sacramento, CA.
- [8] C. T. Chen, *Linear System Theory and Design*. New York: Holt, Rinehart, and Winston, 1984.
- [9] F. M. Salam and G. Erten, "Blind signal separation and recovery in dynamic environments," presented at the IEEE NSIP '97 Workshop, Mackinac Island, MI.
- [10] G. Erten and F. Salam, "Voice output extraction by signal separation," presented at the 1998 Int. Symp. Circuits and Systems, Monterey, CA.
- [11] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. New York: Hemisphere, 1975.
- [12] F. Salam and G. Erten, "Derivation of the state space formulation of blind signal separation and extraction," IC Tech, Inc., Okemos, MI, Internal Rep. 12, 1996.

G. Erten (M'88–SM'97) received the B.S. degree in electrical engineering from Stanford University, CA, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1991 and 1993, respectively.

Between 1985–1991, she was a VLSI Design Engineer with NCR-AT&T, San Diego, CA, where she was responsible for the design and upgrade of two custom NCR9800 IC's, a master co-processor, and memory-interface chip. Following the completion of the Ph.D. dissertation, she co-founded IC Tech, where she serves as President. She has been responsible for supervising a team of engineers and directing several research and development projects in intelligent image sensing, and signal processing and control areas. She has been active in the areas of computer hardware design, image processing, neural networks, and fuzzy logic.

Dr. Erten has served as Vice Chair for the IEEE Southeastern Michigan Power/Industrial Electronics Chapter. She has served on the IEEE Robotics and Machine Vision Subcommittee and was Session Co-Chair for the International Conference on Neural Networks in 1996. In 1997, she was Session Chair for the "Blind Separation of Temporal Signals: Algorithms, Practice and Applications" at the Mid-Western Symposium on Circuits and Systems.

F. M. Salam (F'96) received the B.S. degree from the University of California (UC) at Berkeley in June 1976, the M.S. degree from UC at Davis in December 1979, and the Ph.D. degree from UC Berkeley in June 1983, all in electrical engineering. He also received the M.A. degree in mathematics from UC Berkeley in June 1983.

Since 1991, he has been a Professor in the Department of Electrical and Computer Engineering, Michigan State University, Ann Arbor. He was the Chairman of the Engineering Foundation Conference on Qualitative Methods for Nonlinear Dynamics in June 1986. He has numerous publications in his technical areas of interest, including nonlinear phenomena of circuits and systems, adaptive nonlinear processing, and microelectronic neural systems. He is the co-editor, with T. Yamakawa, of the 1999 Special Issues on "Micro-Electronic Hardware Implementation of Soft Computing: Neural and Fuzzy Networks with Learning," in the *Journal of Computers and Electrical Engineering* (JCEE), and co-editor, with M. Ahmadi, of "Analog and Digital Arrays," in the *Journal of Circuits, Systems, and Computers* (JCSC). He has been Associate Editor of JCSC since 1989 and of JCEE since 1993.

Dr. Salam served as Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS (1985–1987) and IEEE TRANSACTIONS ON NEURAL NETWORKS (1990–1991). He is presently a representative of the Circuits and Systems Society on the IEEE Neural Networks Council. From 1997 to 1998, he was Chairman of the Circuits and Systems Technical Committee on Neural Systems and their Applications.