Unsupervised Analysis of Polyphonic Music by Sparse Coding

Samer A. Abdallah and Mark D. Plumbley

IEEE Transactions on Neural Networks, 17(1), 179-196, January 2006

Abstract

We investigate a data-driven approach to the analysis and transcription of polyphonic music, using a probabilistic model which is able to find sparse linear decompositions of a sequence of short-term Fourier spectra. The resulting system represents each input spectrum as a weighted sum of a small number of "atomic" spectra chosen from a larger dictionary; this dictionary is, in turn, learned from the data in such a way as to represent the given training set in an (information theoretically) efficient way. When exposed to examples of polyphonic music, most of the dictionary elements take on the spectral characteristics of individual notes in the music, so that the sparse decomposition can be used to identify the notes in a polyphonic mixture. Our approach differs from other methods of polyphonic analysis based on spectral decomposition by combining all of the following: a) a formulation in terms of an explicitly given probabilistic model, in which the process estimating which notes are present corresponds naturally with the inference of latent variables in the model; b) a particularly simple generative model, motivated by very general considerations about efficient coding, that makes very few assumptions about the musical origins of the signals being processed; and c) the ability to learn a dictionary of atomic spectra (most of which converge to harmonic spectral profiles associated with specific notes) from polyphonic examples alone—no separate training on monophonic examples is required.

Index Terms

Learning overcomplete dictionaries, polyphonic music, probabilistic modeling, redundancy reduction, sparse factorial coding, unsupervised learning.

Unsupervised Analysis of Polyphonic Music by Sparse Coding

Samer A. Abdallah and Mark D. Plumbley, *Member, IEEE*

***Abstract*—We investigate a data-driven approach to the analysis and transcription of polyphonic music, using a probabilistic model which is able to find sparse linear decompositions of a sequence of short-term Fourier spectra. The resulting system represents each input spectrum as a weighted sum of a small number of "atomic" spectra chosen from a larger dictionary; this dictionary is, in turn, learned from the data in such a way as to represent the given training set in an (information theoretically) efficient way. When exposed to examples of polyphonic music, most of the dictionary elements take on the spectral characteristics of individual notes in the music, so that the sparse decomposition can be used to identify the notes in a polyphonic mixture. Our approach differs from other methods of polyphonic analysis based on spectral decomposition by combining all of the following: a) a formulation in terms of an explicitly given probabilistic model, in which the process estimating which notes are present corresponds naturally with the inference of latent variables in the model; b) a particularly simple generative model, motivated by very general considerations about efficient coding, that makes very few assumptions about the musical origins of the signals being processed; and c) the ability to learn a dictionary of atomic spectra (most of which converge to harmonic spectral profiles associated with specific notes) from polyphonic examples alone—no separate training on monophonic examples is required.**

***Index Terms*—Learning overcomplete dictionaries, polyphonic music, probabilistic modeling, redundancy reduction, sparse factorial coding, unsupervised learning.**

I. Introduction

A. Background and Motivation

THE automatic analysis and transcription of polyphonic music from audio signals is a difficult and interesting problem that has been attracting research attention since 1975 [1] and has yet to find a definitive solution. Researchers working in this area have employed a variety of signal processing methods—including spectral peak detection [2], [3], iterative spectral subtraction [4], [5], networks of adaptive oscillators [6], spectral dictionaries [7], and time-domain waveform dictionaries [2]—for bottom-up data analysis, combined with top-down knowledge-based processing, often using blackboard models to reconcile partial evidence from multiple knowledge sources [2], [8], [9]. The detection of multiple simultaneous pitched sounds has also been studied in the speech processing community, e.g., using banks of damped harmonic oscillators [10]. These systems exemplify, to varying extents, what Cambouropoulos [11] called the "knowledge engineering approach," where the designer explicitly imbues the system with a great deal of his or her domain-specific knowledge, such as expertise with signal processing, psychoacoustic models, music theory, along with heuristic solutions to particular problems.

In this article, we pursue an alternative approach where the balance is more toward "empirical induction," that is, data-driven or unsupervised learning. We take the view that the problem is essentially one of *structure discovery*, whereby we wish to find regularities or patterns in the audio signal. Our hypothesis is that notes are one such regularly occurring structural element, and that a system capable of decomposing its input into structurally significant parts would also be capable of detecting note-like structures in music without requiring the designer to encode his or her prior knowledge about musical sounds.

This approach is closely related to the information-theoretic concepts of efficient coding and redundancy reduction proposed as organizing principles for the sensory systems of biological organisms [12], [13], essentially because structure implies dependency and therefore redundancy: In order to re-represent highly structured sensory data more efficiently, the sensory system must become attuned to the types of structures which tend to occur in a given environment. A corollary of this is that elements of the system or the resulting representation may become associated with specific salient features or objects in the data, since these are the underlying causes of much surface structure.

In recent years, the role of the efficient coding hypothesis in perception, particularly vision, has been the subject of a large and growing body of research (see [14] for a review). Indeed, the very concept of redundancy has been refined and clarified [15] (to the extent that "redundancy management" may be a more accurate way to describe the approach). The emerging consensus is that one of the major functions of perceptual systems is to achieve a *statistical characterization* of sensory data, since this enables prediction, the detection of unusual conditions ("suspicious coincidences" [16]), and the efficient use of representational resources, both material and energetic. Many features of biological perceptual systems can be understood as arising in response to the need to build accurate yet adaptable probability models [17]; in particular, the goals of *factorial* coding—to reduce to a minimum the statistical dependencies between elements in a distributed representation, as in independent component analysis—and *sparse* coding—to build a distributed representation in which relatively few active elements are required to

Manuscript received December 6, 2003; revised March 13, 2005. This research was supported by EPSRC under Grant GR/R19120/01 (Automatic polyphonic music transcription using multiple cause models and independent component analysis) and by European Commission Project IST-FP6-507142 (SIMAC).

The authors are with the Department of Electronic Engineering, Queen Mary, University of London, London E1 4NS, U.K. (e-mail: samer.abdallah@elec.qmul.ac.uk).

Digital Object Identifier 10.1109/TNN.2005.861031

represent a typical scene—have been found to account for many aspects of early vision [18]–[20].

Our system is built around a probabilistic multiple cause model that implements sparse factorial coding [21]–[23]. The observed data, in the form of a short-term Fourier spectrum, is represented as a weighted linear composition of a number of atomic spectral profiles chosen from a larger set, or dictionary. This dictionary is, in turn, adapted to the statistical structure of the data in order to maximize the sparseness of the decompositions and the independence of the weighting coefficients.

For the task of polyphonic music analysis, we assume that the sound at a given point in time is the composition of zero or more instrumental sounds (typically pitched notes) and that only a few of the possible notes are "on" (playing) at any one time. This is reasonable for many styles of music. For example, in a piano solo, typically fewer than 11% out of the available 88 notes will be playing at once. The independence assumption inherent in factorial coding means that we are expecting the note activities to be independent of each other. Clearly, this is not strictly true, (if it were so, we would have random notes and not music) but we can postulate that the notes are *more* independent than, say, the frequency bins of the short-term Fourier transform. It, therefore, seems plausible that a sparse factorial coding system adapted to, e.g., piano solos, could automatically produce a reductive analysis in terms of the underlying notes. There would be no need to define the "note dictionary" beforehand, since the coder would have to discover it automatically in order to find a maximally sparse decomposition of the signal.

Thus, we may summarize our motivations for using sparse coding to analyze polyphonic music, which are two-fold. On the one hand, we have a problem looking for a solution: Many previous approaches were based on relatively "hand-crafted" processing strategies, and only recently have systems based on probabilistic models been proposed [24]–[27]—these enable the use of well defined processes of statistical inference and parameter fitting in response to empirically observed data. On the other hand, we have a solution looking for a problem: Speculation about the role of information theory in the development of perceptual systems has resulted in a methodology, based on the goals of efficient coding and redundancy reduction, which is putatively applicable to a wide variety of signal domains without requiring a lot of domain-specific knowledge to be built into implemented systems—initial applications in audio [28]–[31] have produced promising results, and hence it is interesting to investigate their performance on specifically musical problems. Sparse coding seems particularly suited to the analysis of polyphonic music because of the potential correspondence between the sparse components and the instrumental sounds in the music.

Conceptually similar approaches to polyphonic analysis have been taken by Smaragdis [32], who uses nonnegative matrix factorization [33] rather than sparse coding, Vincent [27], who constructs a more elaborate nonlinear generative model that specifically addresses the analysis of multitimbral polyphonic mixtures, and Plumbley [34], who develops an algorithm for nonnegative independent component analysis (ICA) and applies it to a short polyphonic extract. We present initial results of our subsequent investigations of sparse coding of power spectra using a more accurate multiplicative noise model in [35].

B. Overview of the Paper

In Section II, we review how sparse coding is accomplished within a probabilistic framework. In Section III, we consider the preliminary analysis and preprocessing of the audio into a form suitable for sparse coding, before presenting the results on synthetic and real polyphonic music in Section IV, followed by a discussion and conclusions. Some implementation details, including a novel optimization technique for sparse decomposition and a modification to the dictionary learning algorithm, are given in the Appendix.

In the remainder, boldface lower and upper case symbols will be used to denote vectors and matrices respectively, while italics will denote scalars. Probability density functions will be written as, e.g., $p(x, y)$, $p(x|y)$, or $p(x;\theta)$, which denote, respectively, a joint density, a conditional density, and a density parameterized by θ. The arguments as written will usually be sufficient to indicate which random variables are implied. Expectations of random variables will be written using angle brackets, e.g., $\langle \mathbf{x} \rangle$.

II. Sparse Coding and Noisy ICA

Much of the research into sparse coding has focused on a linear generative model, where the observed data is represented as a weighted sum of elements chosen from a *dictionary* of available features or *atoms*. Within such a framework, a code is sparse if only "a few" nonzero coefficients are required to represent a typical pattern. Field and Olshausen [18] presented an algorithm capable both of performing sparse decomposition in a given dictionary and of adaptively modifying the dictionary to achieve greater sparsity.

A. The Generative Model

A probabilistic interpretation of Field and Olshausen's sparse coder was pointed out both by Olshausen [21] and by Harpur [36]: Underlying it is a latent variable model in which samples of an n-dimensional random vector $\mathbf{x}$ are generated from m independent hidden (latent) variables $(s_1, \ldots, s_m) \equiv \mathbf{s}$ according to

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{e} \tag{1}$$

where $\mathbf{e}$ is a zero-mean Gaussian random vector representing additive noise. Thus, $\mathbf{x}$ is a weighted sum (plus noise) of the columns of the $n \times m$ matrix $\mathbf{A}$ which represents the dictionary of m vectors. The statistical independence of the components of $\mathbf{s}$ implies that the probability density $p(\mathbf{s})$ factorizes

$$p(\mathbf{s}) = \prod_{j=1}^{m} p(s_j). \tag{2}$$

The Gaussian noise model implies that the conditional density $p(\mathbf{x}|\mathbf{s};\mathbf{A})$ is

$$p(\mathbf{x}|\mathbf{s};\mathbf{A}) = \left[\frac{\det \Lambda_{\mathbf{e}}}{(2\pi)^n}\right]^{\frac{1}{2}} \exp -\frac{1}{2}\mathbf{e}^T \Lambda_{\mathbf{e}} \mathbf{e} \tag{3}$$

where $\mathbf{e} = \mathbf{x} - \mathbf{A}\mathbf{s}$ and $\mathbf{\Lambda}_{\mathbf{e}} = \langle \mathbf{e}\mathbf{e}^T \rangle^{-1}$, the inverse covariance of the noise. It is quite straightforward to carry out the analysis with noise of any covariance [30], but if the noise covariance is

known or fixed at an assumed value, equivalent results can be obtained by linearly transforming the input to whiten the noise, such that $\langle \mathbf{e}\mathbf{e}^T \rangle = \sigma^2 \mathbf{I}$. Thus, we can, without loss of generality, work with a scalar inverse covariance $\mathbf{\Lambda_e} = \lambda \mathbf{I}$, where $\lambda = 1/\sigma^2$. The density model for the observed data is obtained by marginalizing the latent variables

$$p(\mathbf{x}; \mathbf{A}) = \int_{\mathbb{R}^m} p(\mathbf{x}|\mathbf{s}; \mathbf{A}) p(\mathbf{s}) \, \mathrm{d}\mathbf{s}. \tag{4}$$

By framing the problem as a statistical model, we can estimate the matrix $\mathbf{A}$ and the latent variables $\mathbf{s}$ from the data in a principled manner. The estimates of $\mathbf{s}$ can then be taken as a representation or encoding of the data. The univariate densities $p(s_j)$ are generally assumed to be super-Gaussian (that is, more strongly peaked and heavy-tailed than a Gaussian) to reflect our prior belief that the source components s_j are "sparse." Indeed, the resulting code will not be sparse unless the prior satisfies certain requirements, namely, that the $-\log p(\mathbf{s})$ is concave/Schur-concave [23].

B. Inference

Given the latent variable model defined by (1), (2), and (3), the two tasks to be addressed are to infer the optimal encoding $\mathbf{s}$ of a given vector $\mathbf{x}$ using a given dictionary $\mathbf{A}$, and to learn a suitable dictionary $\mathbf{A}$ given a sequence of training vectors $(\mathbf{x}_1, \mathbf{x}_2 \ldots)$. Algorithms for both are described in this and the following section.

For fixed values of $\mathbf{A}$ and $\mathbf{x}$, likely values of $\mathbf{s}$ can be inferred from the posterior density

$$p(\mathbf{s}|\mathbf{x}; \mathbf{A}) = \frac{p(\mathbf{x}|\mathbf{s}; \mathbf{A}) p(\mathbf{s})}{p(\mathbf{x}; \mathbf{A})}. \tag{5}$$

MAP estimate is $\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}; \mathbf{A})$, or equivalently

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \mathcal{E}(\mathbf{s}, \mathbf{x}; \mathbf{A}) \tag{6}$$

where the energy function $\mathcal{E}$ is defined as $\mathcal{E}(\mathbf{s}, \mathbf{x}; \mathbf{A}) = -\log p(\mathbf{x}|\mathbf{s}; \mathbf{A}) - \log p(\mathbf{s})$. Under the assumptions of the model, this becomes

$$\mathcal{E}(\mathbf{s}, \mathbf{x}; \mathbf{A}) = \frac{1}{2}\lambda ||\mathbf{x} - \mathbf{A}\mathbf{s}||^2 - \sum_{j=1}^{m} \log p(s_j). \tag{7}$$

The first term can be interpreted as a quadratic misrepresentation cost, the second as a "sparsity cost" (or more accurately, a "diversity cost" [23]) which penalizes nonsparse component activities and depends on the densities $p(s_j)$. A common choice [22], [37] is the l_1 norm $-||\mathbf{s}||_1 = \sum_j |s_j|$ which is equivalent to using a Laplacian prior $p(s_j) = (1/2) \exp -|s_j|$, since in this case, $-\log p(\mathbf{s}) = m \log 2 + ||\mathbf{s}||_1$, and the additive constant $m \log 2$ has no effect on the optimization (6).

In general, an l_α pseudonorm with $0 < \alpha \leq 1$ promotes sparsity [23]. This is equivalent to using a generalized exponential (or generalized Gaussian) prior $p(s_j) = \mathcal{Z}_\alpha^{-1} \exp -|s_j|^\alpha$, where $\mathcal{Z}_\alpha$ is a normalization factor, since

$$-\sum_{j=1}^{m} \log \left(\mathcal{Z}_\alpha^{-1} \exp -|s_j|^\alpha \right) = m \log \mathcal{Z}_\alpha + \sum_{i=1}^{m} |s_j|^\alpha \tag{8}$$

in which $m\mathcal{Z}_\alpha$ is, again, an additive constant independent of the s_j. Note, however, that the posterior $p(\mathbf{s}|\mathbf{x}; \mathbf{A})$ cannot be guaranteed to be unimodal if $-\log p(s_j)$ is concave in any part of its domain, which will be the case for $\alpha < 1$. Under these conditions, not only does the global maximum become harder to find, it also lessens the validity of any approximation of the distribution as a single "lump" of probability mass positioned around the mode. In particular, the Gaussian approximation, originally used to derive the learning algorithm (12) to be given in the next section, breaks down. Examples of Laplacian, generalized exponential, and a "sparsified Laplacian" priors (see (15) in Section IV-A) are illustrated in Fig. 1.

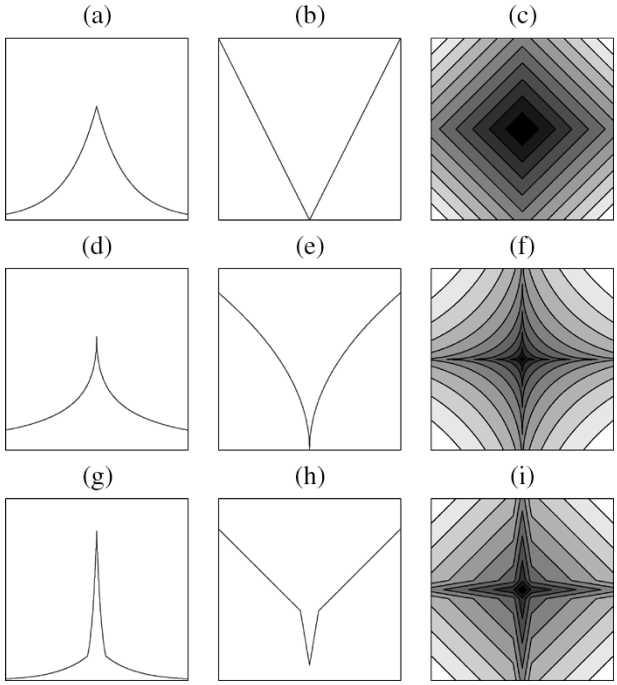

Fig. 1. Three priors for sparse coding. Top row (a–c): Laplacian; middle row (d–f): generalized exponential; bottom row (g–i): "sparsified Laplacian". Left column: density functions, $p(s)$ against s; middle column: negative log-priors, $-\log p(s)$ against s; right column: contours of a two-dimensional diversity cost derived from each prior, $-\log p(s_1, s_2) = -\log p(s_1) - \log p(s_2)$. Black indicates low diversity cost, i.e., preferred sparse configurations.

In the introductory discussion, we described sparse coding in terms of *active* and *inactive* units. Under these terms, a Laplacian or generalized exponential is arguably *not* a sparse random variable at all; indeed, no variable with a finite continuous density can be, since its probability of being exactly zero is infinitesimally small. A truly sparse random variable would have a mixed discrete/continuous distribution, with a finite probability of being exactly zero. Yet, due to a "shrinkage" effect [38], the maximum a posteriori (MAP) inference of continuously distributed components (with sharply peaked priors) can result in a truly sparse code with many exact zeros. Hence, there is a distinction to be made between a sparse *generative model* and model-based sparse *coder*.

For an optimization algorithm well suited to minimizing (6) when using a sharply-peaked prior, see appendix A, where a novel active-set quasi-Newton algorithm is described.

C. Learning

Learning of the dictionary $\mathbf{A}$ can be accomplished by maximum-likelihood estimation: $\hat{\mathbf{A}} = \arg\max_{\mathbf{A}} \mathcal{L}(\mathbf{A})$, where

the log-likelihood function $\mathcal{L}$ over an entire dataset is

$$\mathcal{L}(\mathbf{A}) = \langle \log p(\mathbf{x}; \mathbf{A}) \rangle_{\mathbf{x}} \tag{9}$$

and $\langle \cdot \rangle_{\mathbf{x}}$ denotes an expectation over the training set. The derivative of this function with respect to the dictionary matrix can be shown to be [30], [39]

$$\frac{\partial \mathcal{L}(\mathbf{A})}{\partial \mathbf{A}} = \left\langle \langle \lambda \mathbf{e} \mathbf{s}^T \rangle_{\mathbf{s}|\mathbf{x};\mathbf{A}} \right\rangle_{\mathbf{x}} \tag{10}$$

where the inner expectation is over the posterior density $p(\mathbf{s}|\mathbf{x}; \mathbf{A})$ conditioned on particular values of $\mathbf{x}$. Perhaps the simplest algorithm for optimizing this quantity is that proposed by Field and Olshausen [18], which involves iterative application of the following update:

$$\mathbf{A} \leftarrow \mathbf{A} + \eta \lambda \langle \hat{\mathbf{e}} \hat{\mathbf{s}}^T \rangle_{\mathbf{x}} \tag{11}$$

where η is a learning rate parameter and $\hat{\mathbf{e}} = \mathbf{x} - \mathbf{A}\hat{\mathbf{s}}$. (The functional dependence of $\hat{\mathbf{s}}$ and $\hat{\mathbf{e}}$ on $\mathbf{x}$, due to the inference step (6) has been suppressed to simplify the notation.) Separate renormalization steps must be interleaved with these updates in order to keep the lengths of the dictionary vectors from diverging. In the experiments to be described below, we used an update rule proposed by Lewicki and Sejnowski [22]

$$\mathbf{A} \leftarrow \mathbf{A} + \eta \mathbf{A} \left\langle \gamma(\hat{\mathbf{s}}) \hat{\mathbf{s}}^T - \mathbf{I} \right\rangle_{\mathbf{x}} \tag{12}$$

where the vector-valued function $\gamma : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is the gradient of the (negative) log-prior, $\gamma(\mathbf{s}) = -\nabla \log p(\mathbf{s})$. The Lewicki-Sejnowski algorithm does not require separate normalization steps as the "decay" term (the $\mathbf{I}$ in (12)) keeps the dictionary matrix from diverging. Note that the prefactor of $\mathbf{A}$ means that if the columns of $\mathbf{A}$ do not initially span the full n-dimensional space available to them, they cannot expand into their null space. In these circumstances, it may be necessary to use several iterations of (11) before switching to (12).

D. Overcomplete Dictionaries

When $m > n$, the vectors forming the columns of $\mathbf{A}$ cannot be linearly independent and must form an *overcomplete* dictionary (or degenerate frame). This implies that there are many possible representations of any given vector $\mathbf{x}$. The representation of $\mathbf{x}$ using an overcomplete dictionary is therefore underconstrained. Even if $m < n$, the dictionary may still be overcomplete, in the sense that the vectors may lie within a less-than-m-dimensional subspace, and hence be overcomplete within that subspace, leading again to an underconstrained representation.

By placing fewer constraints on the dictionary vectors, overcomplete dictionaries have a greater flexibility in representing the underlying causal structure behind the observed data: They are able to support multiple interpretations for a given observation, in effect, multiple hypotheses to explain the data. Probabilistic prior knowledge can then be used to determine which is the most likely explanation—effects of this sort are known to play a part in biological perceptual systems [40]. In addition, overcomplete dictionaries have been found to produce lower entropy, more compressible encodings of natural images [39]. Another potential benefit of using a system *capable* of dealing with overcomplete dictionaries is that it should be more robust in the case that the data is noisy and target dictionary is nearly singular (as opposed to actually overcomplete). The performance of ICA on noisy data, for example, degrades more markedly as the noise level increases if the condition number of the target basis matrix is high [41].

III. Preprocessing the Audio

Let us now consider the representation of the audio input. For reasons which we outline below, we chose to work with a phase invariant short-term spectral representation of musical sounds as our source of input vectors $\mathbf{x}$. It is well known that pitched sounds are highly structured in the frequency domain, which suggests that a Fourier analysis would be an appropriate first step. In previous experiments applying ICA to audio waveforms [30], [42], we found that a linear basis well adapted to representing Western classical music is indeed composed of narrowband sinusoids similar to a Fourier basis, at least for short frames of around 50 ms (512 samples at 11.025 kHz). Similar results were also reported by Lewicki [31].

On the other hand, those results [30] also indicate that the phase relationships between different partials of musical notes are generally not consistent enough to allow a linear basis to represent entire note complexes. It may be the case, in certain restricted circumstances, perhaps involving a single instrument and a controlled acoustic environment, that a particular note will always be associated with a particular waveform [43], but environment-dependent variations in the source-to-microphone transfer function, and, for many instruments, the detailed mechanics of sound production, will tend to destroy any phase coherence between partials. Hence, an algorithm to recognize pitch as abstracted from particular pitched sounds must, to some extent, be phase blind, confirming the received wisdom which dates back to von Helmholtz [44].

An alternative view of phase invariance is afforded by a consideration of independent subspace analysis (ISA) and its associated probabilistic model [20]. ISA relates the concept of phase invariant subspaces with clusters of components, with low intercluster dependencies, but higher intracluster dependencies. These intracluster dependencies are modeled by spherically symmetric non-Gaussian probability distributions within the subspace spanned by each cluster of components. Experiments with natural images [20] show that these phase invariant subspaces successfully model a type of shift invariance in visual processing (which is equivalent to a form of phase invariance) and can be found in an unsupervised way. Similarly, in an ICA derived linear basis for music—composed mainly of sinusoidal basis vectors—pairs of components with unrelated frequencies are relatively independent, while pairs of components which differ primarily in phase are strongly dependent and have a circularly symmetric joint distribution [30], [45]. The logical conclusion of this would be to develop an adaptive phase invariant representation using ISA; however, bearing in mind the similarity between the music-adapted ICA basis [42] and a Fourier basis, the experiments presented here are based on the short-term Fourier transform magnitude (STFTM), corresponding to a fixed Fourier basis and fixed two-dimensional phase invariant subspaces.

In contrast with other researchers who have also used the STFTM as input to ICA-based systems [28], [29], [46], we do not use principal component analysis (PCA) to reduce the dimensionality of the input spectra, since this presupposes that information in the spectrogram relevant to polyphonic pitch detection is associated with high variance components. As a counterexample, consider a hypothetical case where some low-magnitude upper harmonics of a instrumental sound are well defined and supply accurate pitch information, while the low frequency part of the spectrum is masked by loud nonpitched sounds—dimensionality reduction using PCA would retain the high-variance low-frequency components and discard the informative upper bands. Hence, we prefer to work with the full spectrum, allowing the system to "decide" where the information is.

A. Initial Statistical Characterization of Spectra

Our first step after the STFTM is to perform a statistical characterization of the spectra. Previous investigations into the statistical structure of natural sounds have shown that, under many representations, they tend to be extremely non-Gaussian. For example, a PCA-based spectral analysis of television broadcasts [47] showed that the PCA coefficients (broadly comparable to Fourier coefficients) had sharply-peaked, heavy-tailed distributions.

Below, we describe how a simple parametric model can be used to obtain an initial characterization of the STFTM, as well as providing a model-based framework for preprocessing. Letting y_i denote the magnitude of the ith complex Fourier coefficient, we assume a generalized exponential density model

$$p(y_i) = \frac{w_i \exp -(w_i y_i)^{\alpha_i}}{\alpha_i^{-1}\Gamma\left(\alpha_i^{-1}\right)}, \qquad y_i \geq 0 \tag{13}$$

where Γ is the gamma function, $\Gamma(t) = \int_0^\infty x^{t-1}e^{-x}\mathrm{d}x$. The parameters $w_i > 0$ and $\alpha_i > 0$ characterize the scale and super-Gaussianity respectively of each spectral band and are fitted to the data using the maximum-likelihood criterion (see [48] for further details and [49] for an online natural gradient algorithm). Fig. 2 illustrates the results of fitting these parameters to an ensemble of 256-band short-term spectra obtained from a set of eight piano pieces lasting approximately one hour in total. The fitted distributions are strongly super-Gaussian with most of the exponents α_i around 0.5 (a Gaussian distribution would have $\alpha_i = 2$, and a Laplacian, $\alpha_i = 1$). In addition, there is a noticeable oscillatory variation of α_i with i and hence with frequency. A similar experiment with a much longer data set (several hours of live music radio) confirms that these oscillations occur at 12 cycles/octave, reflecting the tuning of the Western scale with 12 semitones per octave. The generalized exponential model fits many but not all of the frequency bins (see Fig. 3 for two examples).

In the above analysis we treated each spectral band independently. However, it is precisely the dependencies between spectral bands that the sparse coder is intended to exploit; indeed, the presence of recognizable patterns in the spectrogram implies that these dependencies exist. A pair-wise correlation analysis begins to indicate where the dependencies lie: The cross-correlation matrix for the piano spectral data is illustrated in Fig. 4(a). The straight lines at various slopes imply correlations

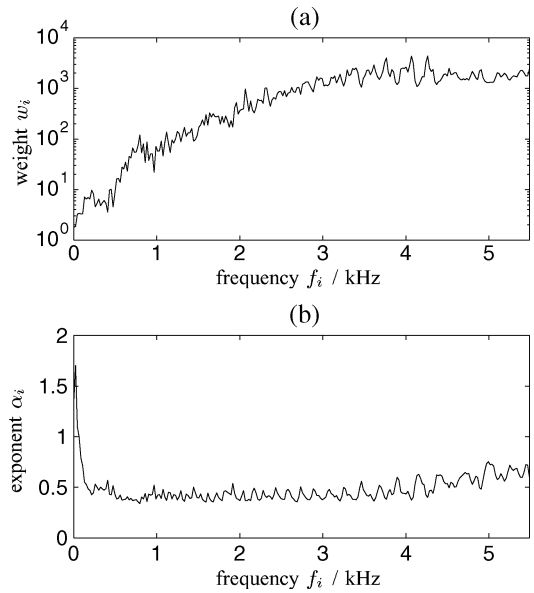

Fig. 2. Parameters of generalized exponential density model (13) fitted to piano STFTM data.

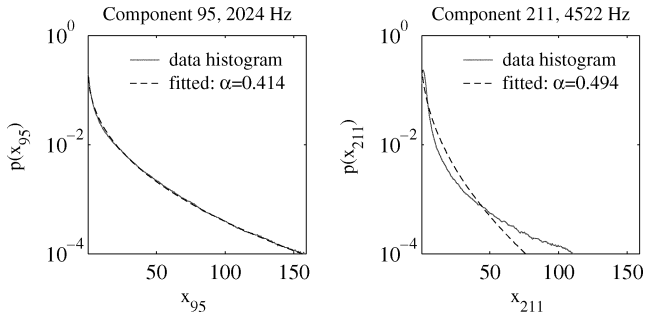

Fig. 3. Comparisons between histograms and fitted densities for two components. The first is an example of a well fitted histogram, the second is less well fitted. In general, the components above around 4 kHz have heavier tails but more rounded peaks than the fitted generalized exponentials.

between harmonically related frequencies, and once again, there is some 12 cycle/octave variation. Fig. 4(b) is the cross-correlation matrix obtained from a much longer and more varied sample and shows both theses structures more clearly. Note that, though this second-order analysis is sufficient to confirm that there are dependencies between spectral bands, it is not enough to characterize them completely, which is why methods (such as PCA), which rely on second-order statistics only, are incapable of finding sparse factorial representations.

B. Noise Model

For correct operation of the sparse coder, it is important that the input conforms to the assumed Gaussian noise model (3) as well as possible. This is not an accurate noise model for the STFTM: The additive noise in the signals we examine is very low, and what appears to be "noise-like" activity in the spectrum is actually due to interactions between different components in the signal and the STFT window—so-called "spectral leakage" [50]. These effects result in what is best considered a multiplicative noise model, where the random (or at least unmodeled) variations are proportional to the activity in a local region of the spectrum.

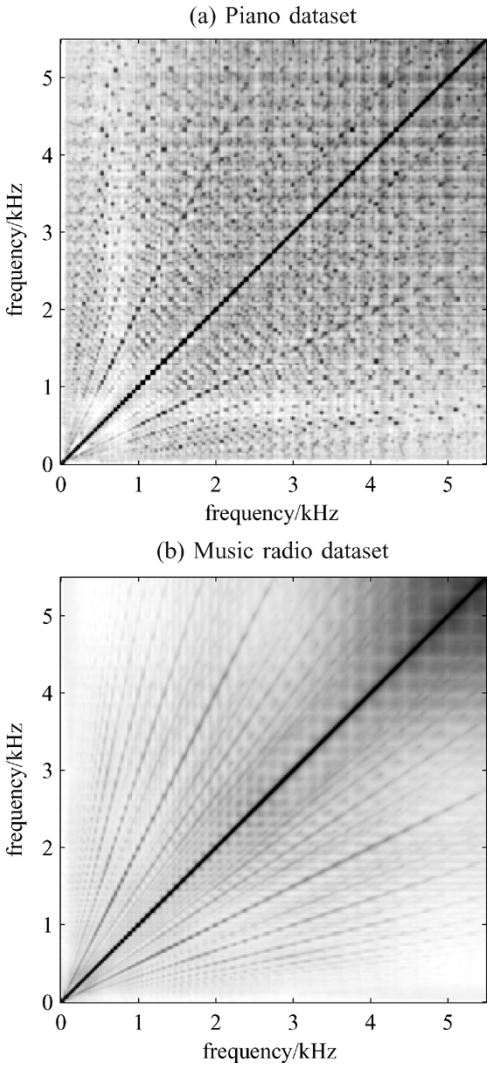

Fig. 4. Cross-correlation of spectral magnitudes for (a) one-hour piano dataset and (b) several hours of music radio. The straight lines indicate correlations between harmonically related frequencies.

To implement the sparse coder, this multiplicative noise must be approximated as an additive noise of fixed variance. As a first approximation, we assume a diagonal noise covariance with a standard deviation at each frequency proportional to the average spectral magnitude at that frequency; that is, inversely proportional to the fitted values of w_i in (13). Hence, to whiten the noise as described in Section II-A, we normalize the STFTM components y_i by the scale factors w_i, i.e., the input to the sparse coder is $\mathbf{x}$ such that $x_i = w_i y_i$. The scalar noise variance $\sigma^2 = 1/\lambda$ is left as a free parameter to allow the sparse coder's performance to be tuned.

Spectral flattening of this sort, sometimes known as "pre-emphasis," is a common tool in signal processing. In the context of sparse coding, it has a rather precise motivation: To sphere the effective noise distribution.

C. Loudness Compensation/Gain Control

One of the problems of dealing with recordings of live music is the large dynamic range caused by the alternation of loud and soft passages, or of loud and soft pieces. It was empirically observed that batched-online learning was generally more stable at higher learning rates if these fluctuations were attenuated by roughly normalizing the loudness, at least over time-scales greater than the batch size. To this end, an adaptive gain control was implemented within the model (13) by factorizing the w_i (introducing an explicit time dependence) as

$$w_i(t) = B(t)\beta_i \tag{14}$$

where the β_i are assumed fixed, but $B(t)$ is a time-varying overall scale factor. An online maximum-likelihood estimate of $B(t)$ is maintained using stochastic gradient ascent with a learning rate fast enough to track changes over a time-scale of the order of 5 s. Note that, because of the generalized exponential density model, $1/B(t)$ effectively measures loudness in the spectral domain using a generalization of an l_α pseudo-norm, in which each dimension (frequency bin) has its own exponent α_i and weighting β_i determined adaptively from the data. Also note that, because $B(t)$ is shared across all frequencies, it can be estimated using much less data, and therefore tracked more quickly, than is the case for the β_i.

IV. Sparse Coding of Polyphonic Music

A. Synthetic Harpsichord Music

We first present results obtained from an analysis of music generated synthetically from a MIDI encoded version of Bach's *Partita in A minor, BWV827*, which was chosen because it consists mainly of two or three relatively independent voices with few block chords. The MIDI data was rendered using the harpsichord patch of the wavetable synthesizer built into a PC sound card. The harpsichord sound was chosen because it is harmonically quite rich and the temporal evolution of each note consists primarily of a decay without frequency modulation or much change in the balance of the harmonic components. The synthesis was carried all the way to an analogue continuous time signal, which was then sampled at 11 025 Hz and digitized at 16 bits per sample using the same sound card. The audio frames used to generate the STFT were 512 samples long (46 ms) with a 50% overlap. The magnitudes of the first 256 bins of the Fourier transform were retained to form the 256-dimensional input vectors $\mathbf{x}$.

The generalized exponential model-based spectral normalization described in Section III-A was not used; instead, a simple amplitude-based dynamic gain control was used to normalize the signal level (in the time domain) over a time-scale of 5 s.

Motivated by a desire to model the expected sparsity of the code, a "sparsified Laplacian" prior was used, which is a piece-wise exponential approximation to a sparser-than-Laplacian density [see Fig. 1(g)–(i)], and is specified as

$$p(s) = \begin{cases} \mathcal{Z}_{\mu\rho}^{-1} e^{-|s|} : & |s| \geq \mu \\ \mathcal{Z}_{\mu\rho}^{-1} K e^{-\rho|s|} : & |s| < \mu \end{cases} \tag{15}$$

where $\mathcal{Z}_{\mu\rho}$ is a normalization constant, and $K = e^{\mu(\rho-1)}$ to ensure continuity. The parameters $\mu \geq 0$ and $\rho \geq 1$ control the width and relative mass of the central peak.

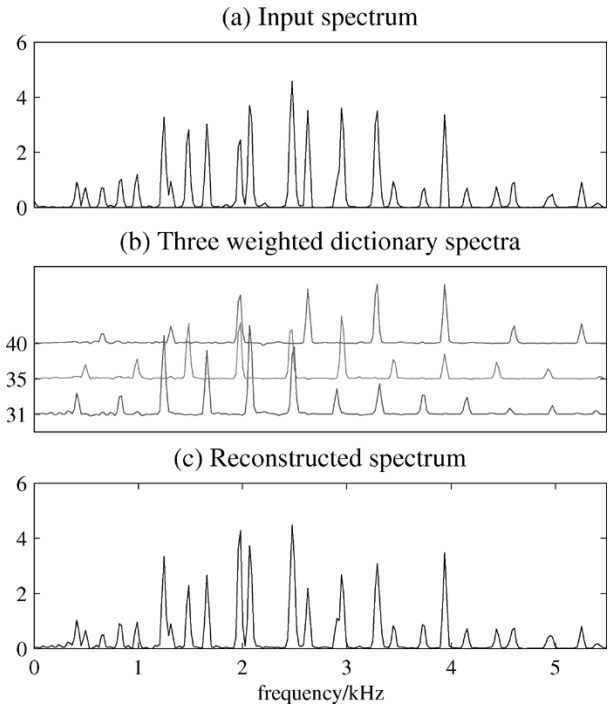

Fig. 5. Decomposition of a three-note chord spectrum (a) into a weighted sum of note spectra selected from the dictionary illustrated in Fig. 6(a). The decomposition was the obtained using the inference method described in Section II-B. The y-axis labels (b) indicate which dictionary elements are active in the decomposition. The reconstructed spectrum (c) is obtained by summing the weighted dictionary spectra.

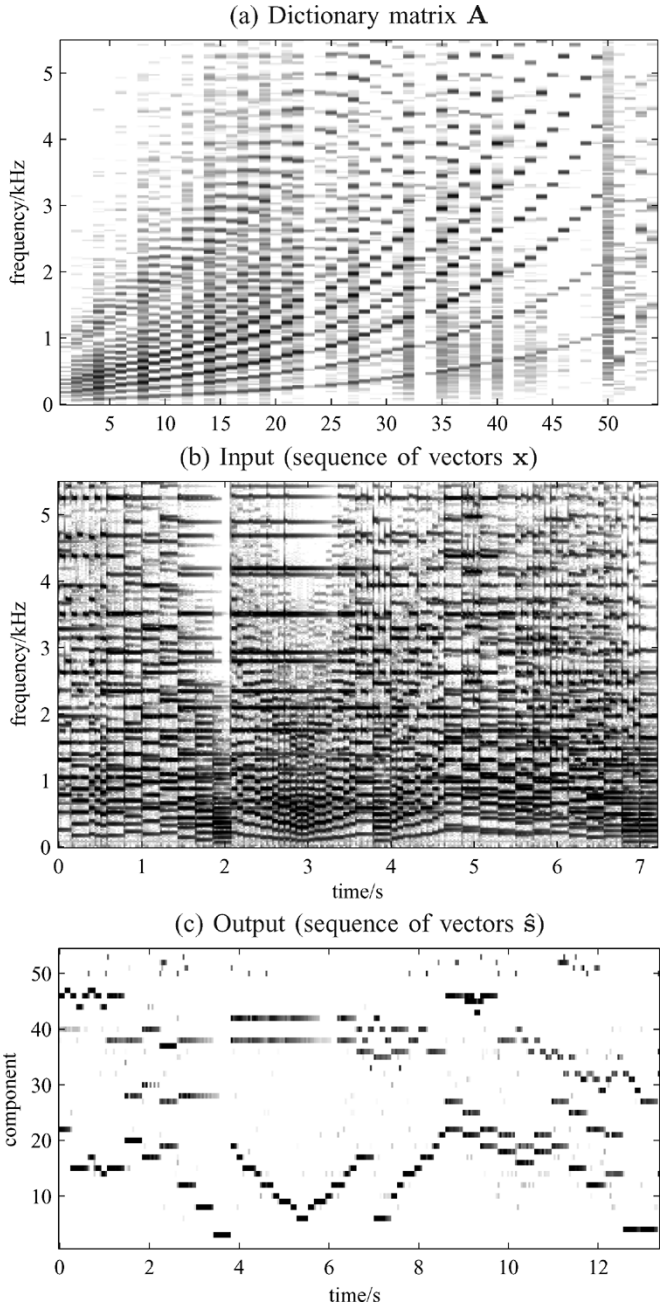

Fig. 6. Harpsichord sparse coding results. (a) 54 nonzero basis vectors obtained from sparse coding synthetic harpsichord music (each column of the matrix is one dictionary vector). (b) Input to and (c) output from sparse coder using the above dictionary (these are just short extracts, not the complete data.) The images (a) and (b) are on a logarithmic grey scale with a dynamic range of 24 dB. The first 49 dictionary vectors represent pitches from B1 to C6; the remaining five were judged to be of indeterminate pitch.

The active-set optimizer described in Appendix A was used to do the inference step, notwithstanding the fact that the sparsified Laplacian prior might result in multiple local optima.

After some initial experimentation, the model noise parameter σ was set to 0.2, and the parameters of the sparsified Laplacian prior were set to $\mu = 0.05$ and $\rho = 24$, the aim being to balance the sparsity of the decomposition (and the speed of the inference step using the active-set optimizer) with the accuracy of the spectral reconstruction $\mathbf{As}$. A typical spectrum, along with its decomposition and reconstruction, is shown in Fig. 5. A 256×96 dictionary matrix $\mathbf{A}$ was initialized to the first 96 columns of the 256×256 identity matrix. An initial phase of learning was carried out using the update rule (11) to allow the 96 dictionary vectors to explore the 256 dimensional input space. The remainder of the learning was carried out using the update rule (12). Despite the initial dictionary being undercomplete, many of the vectors decayed to zero, leaving 54 nonzero vectors. The sounds corresponding to each of the dictionary vectors were synthesized by filtering white noise with finite impulse response (FIR) filters matching each of the spectral profiles. All but seven were judged to produce pitched sounds with pitches from D2 to C6, (where C4 is middle C at 261 Hz) and were manually reordered accordingly. Of the remaining seven, two were found to respond reliably to the presence of the lowest pitches B1 and C2, for which the harmonic spacing is at the limit of the frequency resolution afforded by the STFT. The resulting dictionary matrix is illustrated in Fig. 6(a).

Note that the assessment of the pitchedness and pitch of each dictionary element was necessarily a subjective evaluation since there is no precisely defined test of whether or not a sound is pitched—pitch is a psychological quality, and as such the final arbiter of pitch is human perception.

The output from the sparse coder, that is, the sequence of MAP estimates $\hat{\mathbf{s}}$, is illustrated in Fig. 6(c). Over the entire 11 minute dataset, the 54 active components, on average, are 95% of the time exactly zero. Histograms of some of the individual note activations are shown in Fig. 7: Most of the histograms are somewhat bimodal, suggesting (accurately in this case) an

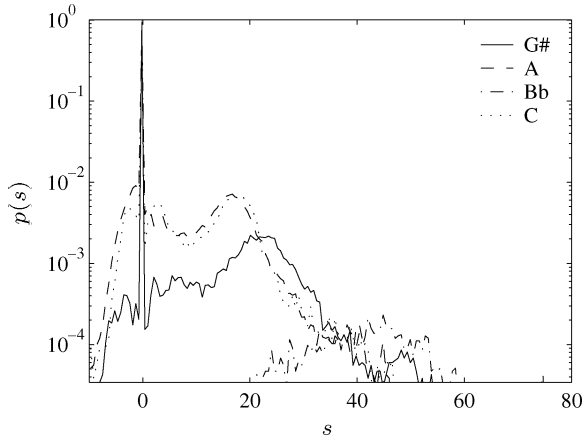

Fig. 7. Histograms of the activities of a few of the sparse components (labeled by pitch) obtained from the harpsichord dataset, showing both high sparsity (the large spikes at zero) and some bimodality, reflecting the underlying binary note activations. The relative widths of the distributions are related to the lengths (in the Euclidean sense) of their corresponding dictionary vectors. These in turn are partly determined by the frequency of occurrence of each note, as suggested by an analysis of the learning algorithm [30]. Hence, in this example, the B♭ distribution is widest because B♭ is rarely used in the key of A minor.

underlying binary process, namely, the presence or absence of a note.

To evaluate the potential of the sparse coder as part of a transcription system, a very simple threshold-based MIDI event trigger was implemented. For each pitched dictionary vector, a note-on event is triggered whenever its corresponding activity $\hat{s}_j$ rises above a fixed threshold for two consecutive time-slices, and a note-off event is triggered when the activity drops below that threshold for two consecutive time-slices. The activities of the nonpitched dictionary elements were ignored. The MIDI sequences[1] were then resynthesized using a piano patch (which is somewhat less painful to listen to than the harpsichord patch used to synthesize the original). Subjectively, the generated MIDI sequence was a generally accurate though sometimes rhythmically imprecise reproduction of the piece.

A quantitative evaluation was made by matching the note-on events for each pitch to those in the original MIDI file, to within a tolerance of about 50 ms. Of the 4684 onsets in the evaluation set, 94.3% were correctly detected, while 2.2% of the 4515 triggered onsets were "false positives" that did not match any note in the original. For comparison, Marolt's transcription system [6] achieves, for synthesized music, average detection and false positive rates of 90% and 9%, respectively, improving to 98% and 7% for a synthesized Bach partita similar to the one we used. Klapuri [51] quotes an "error rate" of between 4% for two note chords, rising to 12% for four note chords.

In view of the relatively unsophisticated event triggering scheme and the lack of temporal integration in the sparse coder itself, some "repeated note" errors are to be expected. These occur when spurious onsets are generated during a sustained note, or conversely, repeated *legato* notes are coalesced into one long note. If such errors are not counted, then the detection rate goes up to 99.2% with a false positive rate of 1.0%. By integrating better onset detection algorithms into the system [52], [53] we may reasonably expect to approach this level of performance.

[1]Available, along with the synthesized dictionary vectors, at http://www.elec.qmul.ac.uk/department/staff/research/samer.htm.

B. Improvements in Dealing With High Sparsity

The results of the experiment with harpsichord music (and other experiments with a toy system not shown here, see [30]) indicate that, despite the use of a "sparsified Laplacian" prior, the algorithm as it stands does not cope well with very sparse underlying components: In such cases, the learned dictionary vectors are incorrectly scaled, converging to a fraction of their correct lengths (hence the systematic variation in the widths of the histograms in Fig. 7, or they decay away completely. Any attempt to produce a "cleaner" encoding, with more exact zeros for absent notes, either by adjusting the prior parameters or increasing the model noise parameter σ, tends to exacerbate this problem. The quality of the harpsichord transcription was not seriously affected because the instrumental tone was so simple, requiring only one dictionary vector per note, but preliminary experiments with real instrument sounds showed that too many dictionary elements decay away completely to enable the others to represent the variations in the spectra of all the notes. For example, when piano recordings (to be described in Section IV-C) were analyzed with the model noise parameter set at $\sigma = 5$, (i.e., the same as in Section IV-C) but without the "decay when active" correction (described later), all but 77 out 256 dictionary vectors decayed to zero, and several pitches in the lower octaves were not represented at all. When the noise parameter was doubled to $\sigma = 10$, only 23 nonzero dictionary vectors remained.

An analysis of a simplified form of the system [30] suggests why this might be the case: The approximations made in deriving the algorithm are not accurate for truly sparse priors, that is, ones that have a finite probability mass at zero, resulting in systematic errors in dictionary learning. This can be addressed by introducing a more complex generative model with discrete latent variables [27], [54], but an alternative is to make a small modification to update step (12), to the effect that the decay term in (12) is applied only to those columns of the dictionary matrix whose corresponding components are active (see Appendix B for a derivation).

This "decay when active" correction means that none of the dictionary vectors decay completely even if they are very infrequently active, making the system as a whole more robust when the signal-to-noise ratio is low. It also means that the scaling of the sparse components s_j is such that their marginal distributions (not shown) are fitted to a Laplacian density *without* being affected by the "large spike" at zero (see Fig. 7); that is, only the "active" part of the distribution is normalized to unit mean absolute deviation.

C. Real Piano Music

The experiments with harpsichord music were intended primarily as a "proof of concept," and show that a system based on adaptive sparse decomposition can indeed perform a meaningful analysis of polyphonic mixtures. The synthetic harpsichord music, however, is rather unrealistic in that there is little or no variation between different instances of the same note, and little variation in dynamics over the course of the piece.

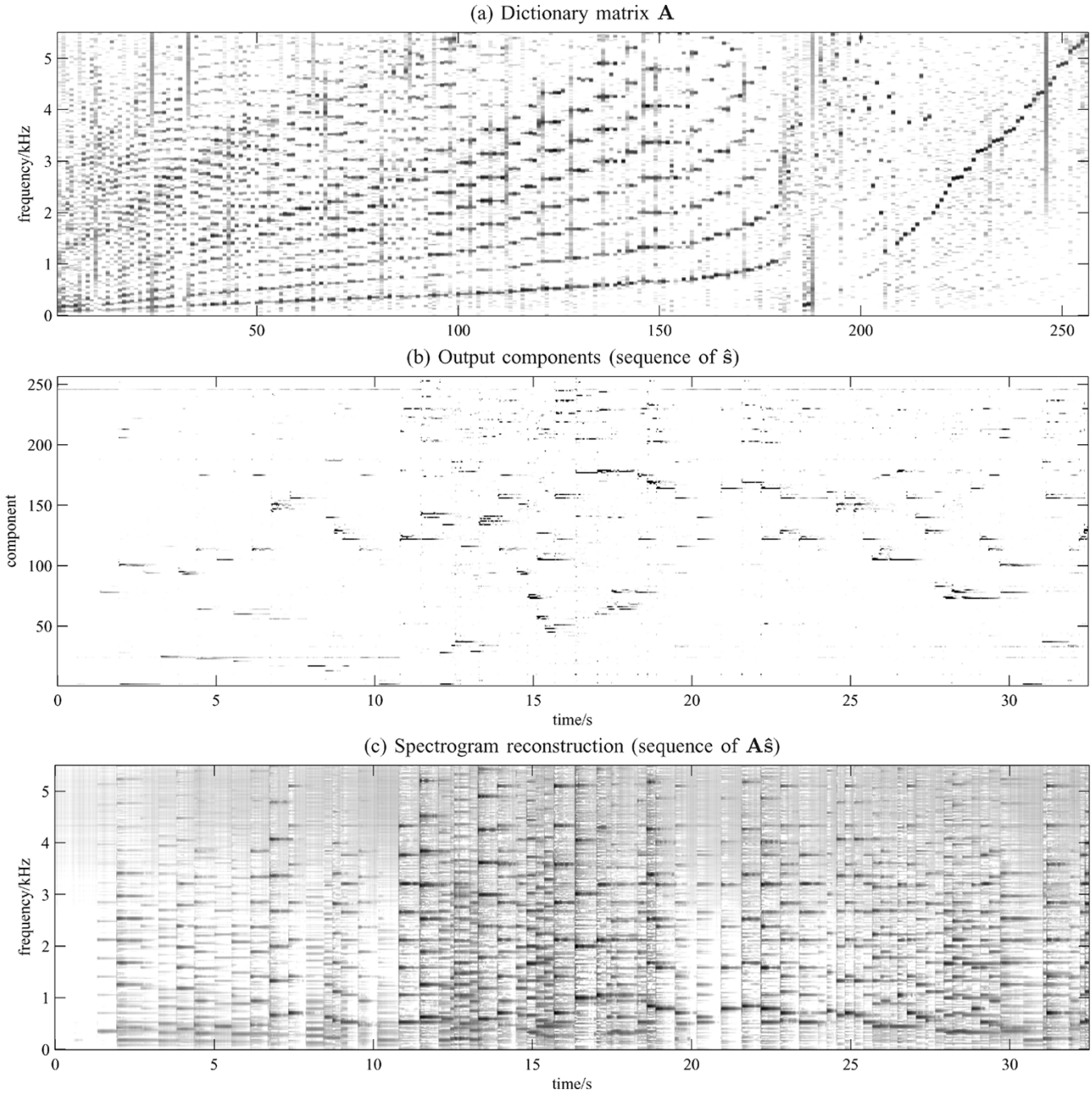

Fig. 8. Piano results. (a) Dictionary matrix, reordered by estimated pitch. (b) Sparse components $\hat{\mathbf{s}}$. (c) Reconstruction of inputs $\mathbf{x}$ from sparse components. The extracts (b) and (c) are from the opening from Bach's three part invention, no. 9 in F minor.

To investigate the performance of the sparse coder with more realistic sounds, a 256-element dictionary was trained on one hour's worth of acoustic piano recordings (Bach's two and three part inventions performed by Andras Schiff, plus fugues 14–19 from *The Well Tempered Clavier, Book I* performed by Jeno Jando). The model-based spectral normalization described in Section III-A was used to preprocess the sequence of STFTM spectra, while a Laplacian prior was used in the sparse coder, along with an additional "decay when active" correction (see Section IV-B and Appendix B). The noise parameter was set to $\sigma = 5$, which should be compared with the distributions of the pre-emphasised spectral components illustrated in Fig. 3. The 256×256 dictionary matrix $\mathbf{A}$ was initialized to the identity and adapted according to (12) using online updates computed from batches of 256 frames. The learning rate η was set to 0.005 initially, then increased to 0.02 after 1000 updates, then gradually reduced to 0.001.

The dictionary after approximately 5000 updates is illustrated in Fig. 8(a). A visual and auditory inspection indicated that 179 of the dictionary vectors correspond to pitched spectra or fragments of pitched spectra. Their fundamental frequencies were estimated manually by matching against an adjustable harmonic ladder. We noticed a slight but consistent inharmonicity, with the upper partials stretched with respect to the

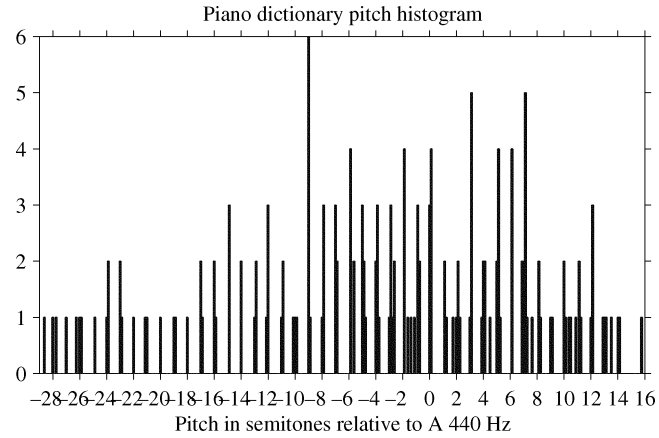

Fig. 9. Distribution of fundamental frequencies (on a logarithmic scale) of spectra in the piano derived dictionary. The frequencies were estimated by fitting parameterized inharmonic sequences [55] to the peaks extracted from each spectrum.

fundamental; this is a known property of piano tones [55]. Subsequently, the fundamental frequencies of the dictionary vectors were re-estimated by a parametric fitting of inharmonic spectra. The resulting distribution of fundamental frequencies, on a logarithmic scale, is illustrated in Fig. 9. They are, to a great extent, clustered to a 12-degree-per-octave scale, forming 45 well defined pitch equivalence classes covering over three octaves from E2 to C6. Within each pitch class, there are several dictionary vectors which, between them, define a low-dimensional spectral subspace (see Fig. 10). Thus, when a note is sounded, one or more members of the associated class will tend to become active in the sparse decomposition, as illustrated, for example, in Fig. 11. The presence or absence of that note can therefore be assessed in terms of the total activity in the subspace. The activities of each pitch class over a 30 s extract are illustrated in Fig. 13.

A complete transcription system based on the sparse coder would require a more intelligent event detection stage than the simple scheme used for the harpsichord music. We aim to apply a recently developed onset detection algorithms [52], [53] to the per-pitch class activity traces, but this has yet to be implemented; until then, a direct quantitative comparison between this system and other music transcription systems is not possible.

D. Effects of Preprocessing

To assess the effect of the spectral normalization step, or "pre-emphasis," on the dictionary learning and sparse decomposition (see Section III-B), the experiment with piano data was repeated using the raw STFT magnitudes as input, exactly as in the harpsichord experiment. The noise parameter was set to $\sigma = 0.2$, which, by direct observation, is smaller than the random variations in the spectral magnitudes at low frequencies, but much larger than those at high frequencies.

The dictionary matrix was initialized to the 256×256 identity. After 2000 updates, only the first 120 or so dictionary vectors changed significantly. The remaining dictionary vectors were essentially unaltered, with their corresponding components in the sparse decomposition active only during very loud and percussive onsets. In addition, the upper half of the spectrum, from about 3 kHz up, was very poorly reconstructed from the sparse decomposition, with clearly resolved harmonics in the input spectrum not visible in the reconstruction.

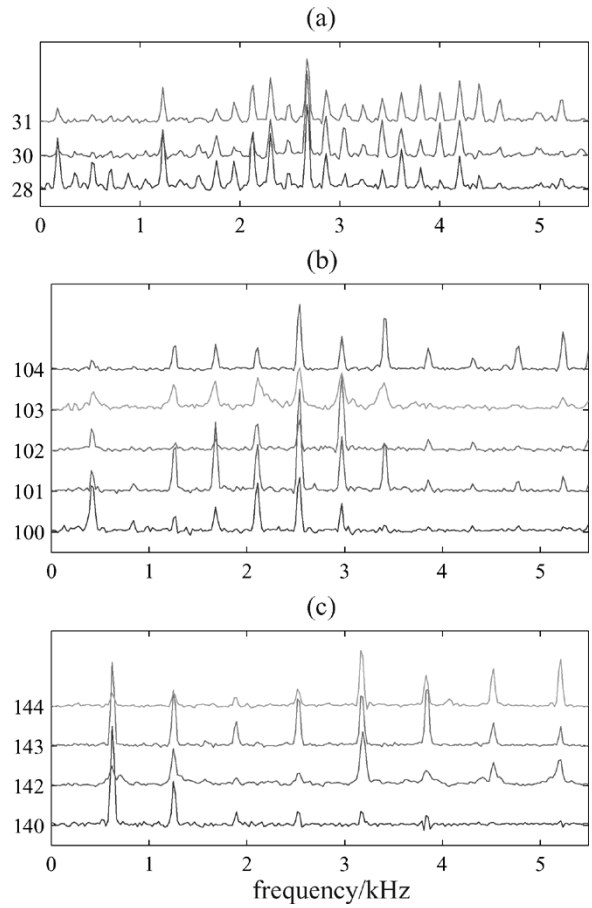

Fig. 10. Three of the pitch equivalence classes formed by the columns of the piano-based dictionary matrix. In each group, the spectra have the same pitch but a different balance of harmonics, enabling the group to model different instances of the same note, or different stages in the evolution of a single note. Each dictionary vector is labeled by its index in the dictionary. (a) F3 group. (b) $A\flat 4$ group. (c) $E\flat 5$ group.

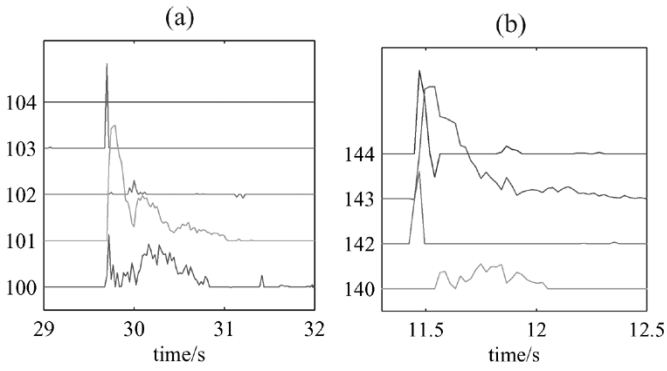

Fig. 11. Illustration of how the components in a pitch class collectively encode a single note. Two notes, corresponding to the dictionary vector groups shown in Fig. 10(b) and (c), were extracted (preserving the time axis labels) from the encoding of the sequence shown in Figs. 8 and 13. (a) $A\flat 4$ group. (b) $E\flat 5$ group.

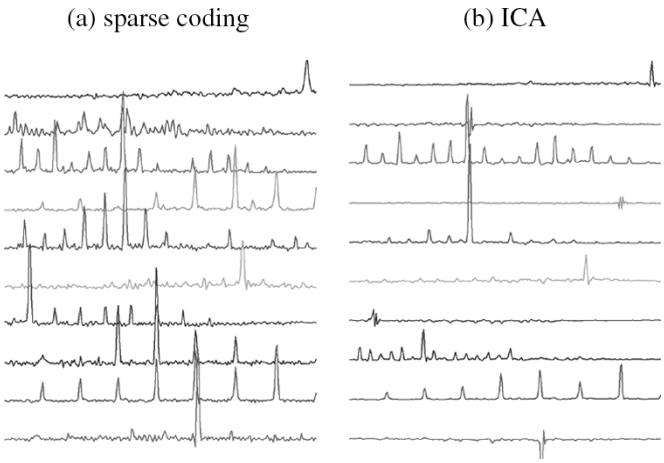

Fig. 12. Comparison between ten elements chosen at random from (a) sparse coding dictionary and (b) an ICA basis, both learned from the same piano dataset. Both systems were initialized with the identity matrix, so in many case, the corresponding elements represent similar collections of frequency components, but the sparse coding dictionary goes much further toward representing entire note spectra.

E. Using ICA Instead of Sparse Coding

A final experiment was conducted to assess whether or not sparse coding yields any significant improvement over standard (noiseless) ICA. A standard online ICA algorithm [56] was applied, using the same spectral normalization and Laplacian prior as were used as in the piano experiment. The 256×256 weight matrix was initialized to the identity, and after approximately 10 000 updates, the weight matrix was inverted to yield 256 basis vectors which can be compared with the dictionary vectors produced by the sparse coder. Though it is difficult to illustrate a representative sample of the basis vectors, Fig. 12 compares ten vectors chosen at random from the sparse coding dictionary and the ICA basis, and gives some sense of the difference in character between them. (The same columns were taken from both matrices and in many cases have roughly the same form.) Though many of the ICA basis vectors did take on some harmonic structure, the effect is generally less pronounced, especially for low pitches. The overall effect is that it is not possible to identify individual pitches with small, nonoverlapping sets of components to the same extent as was possible with the sparse coding dictionary. ICA does produce a set of coherent spectral features, but most of them cannot be directly interpreted as "note detectors", since each note produces a much more distributed activity pattern than in the sparse coder, and the activity patterns for different notes overlap.

As a further confirmation of the extent to which the sparse coder is able to concentrate its representational capacity, the condition number of the ICA basis is 53, while that of the sparse coding dictionary is 8.1×10^8. Indeed, the effective rank of the dictionary matrix (judged from a sharp "elbow" in its singular value spectrum) is 122, which means that the 256-element dictionary is essentially two-times overcomplete within a 122-dimensional subspace.

V. Discussion

The sparse coding system we are using was motivated by very general considerations hypothesized to account for all types of perceptual organization, and is basically the same as those used as in current models of early visual perception [18], [22] with very few modifications addressed at analyzing specifically musical sounds. Rather than incorporating physiologically based auditory models, psychoacoustic data, or prior musical knowledge, the system is driven by statistical regularities in the data, developing the relevant processing strategies over the course of learning. For example, one might say that it has discovered the well known principle of grouping by harmonicity [57] for itself. The main design decisions which might have prejudiced the outcome are the choice of a short-term spectral input representation with certain time and frequency resolutions. However, even these choices are supported by experiments with ICA of musical waveforms, which suggest that a phase invariant time-frequency representation of this kind could be arrived at in a purely unsupervised way in response to the statistical structure of musical sound, in the same way that independent subspace analysis has been found to account for the shift-invariant properties of the complex cell responses in visual cortex [20].

Hence, these experiments, along with similar work with other types of data [28], [29], [58] lend support to the hypothesis that unsupervised learning in statistical models, motivated by the efficient coding hypothesis as discussed in Section I-A, is a viable computational model for perception in several sensory modalities.

As far as musical applications are concerned, our approach has two advantages over other nonmodel-based spectral dictionary methods [5], [7]: First, the decomposition is formulated as a statistical inference problem, optimizing an explicitly given cost function defined in terms of an empirically verifiable probability model, rather than a heuristically defined procedure, perhaps one not even optimizing any particular cost function as in the case of greedy algorithms like matching pursuit. Second, the note dictionary itself can be learned from polyphonic training data, rather than requiring an initial analysis of specially prepared monophonic data.

A. Partial Discretization and Object Based Coding

Consider a vector quantizer or discriminative classifier: Each input vector is encoded by *identity* of one active unit chosen from a finite or countable collection. This is a purely discrete representation. On the other hand, the elements of a sparse code can be either *off* or *on*, taking some value from a continuous range, and while multiple elements may be active together, these will usually be few in number. Such a representation can be interpreted as a variable length, but usually short, *list* of discrete entities or objects, each of which is associated with, or parameterized by, one continuous attribute, and echoing the concept of object-based coding in communication theory (see [59] for a music-related example).

Thus, we can think of sparse coding as a sort of "partial discretization," forming an intermediate stage between continuous data and a fully discrete, or symbolic, representation, such as a musical score or a categorical percept such as pitch. Certainly, this was one of our motivations for using sparse coding, since our hypothesis was that the notes would emerge as perceptual objects. Under the synthetic harpsichord dictionary, the inactivity of a component was indeed a 99% reliable indicator of the

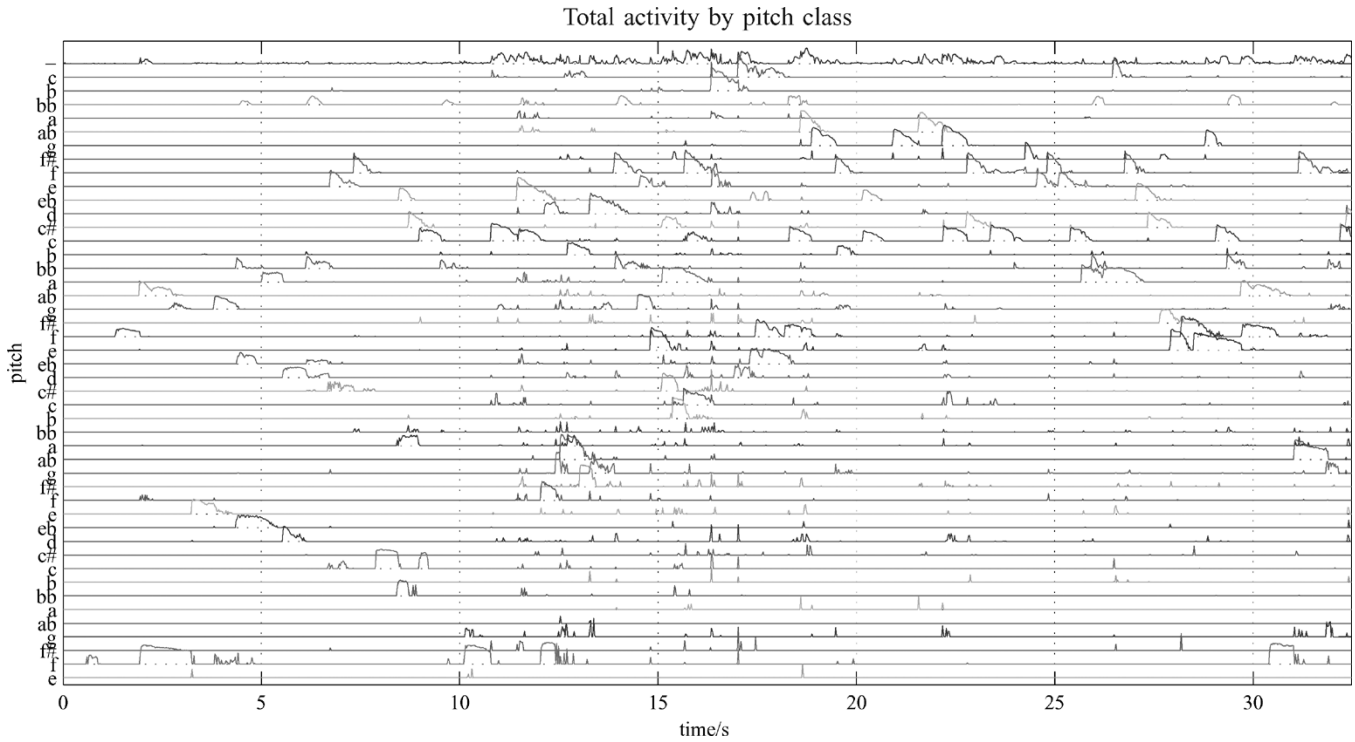

Fig. 13. These traces show the total activity in each pitch equivalence class, computed by summing the squared activities of the components in the group. The traces themselves are plotted on a logarithmic y-scale, with a dynamic range of about 40 dB, and labeled by pitch. The top trace is the total activity of all the unpitched dictionary vectors.

absence of a note, and though the converse was not true—the activity of a component was not, without further thresholding, a reliable indicator of the presence of a note—the use of a sparse coding model with explicit discrete latent variables [54] is likely to yield an almost exact correspondence between active components and sounding notes.

The situation with the piano dictionary is quite different. The variability of the piano tones and the fact that the "decay when active" correction allowed more dictionary vectors to be retained meant that each note came to be associated with a group of components, so that any event triggering would have to be based on an assessment of the activity of the whole group. In addition, both dictionaries occasionally result in negative component activities (see the marginal component histograms of Fig. 7): These indicate that a dictionary element is being used, not because the note it represents is present, but to refine the reconstruction of an input spectrum which is largely but not completely accounted for by other dictionary elements. All of this calls into question the utility of an explicit *on/off* coding for individual components, and suggests that a form of "sparse subspace coding," where the diversity cost depends on the number of active subspaces rather than components, would be a worthwhile avenue for further investigation.

B. Comparison With ICA

We have already commented that music transcription involves a transformation from continuous or analogue data to a discrete or symbolic representation. This necessarily involves a loss of information, whether it is the rejection of what is held to be "noise", or the formation of categories, within which any random variations are ignored. Sparse coding takes a step toward this by including an explicit noise model (which means that denoising is an intrinsic part of the inference process) and by achieving the sort of partial discretization discussed above.

In contrast, ICA does not involve any loss of information and does not achieve partial discretization. Even ICA algorithms driven by what are commonly adopted as "sparsity measures" (e.g., kurtosis) do not, in general, produce encodings with exact zeros, because with a complete basis and no noise model, each component is a fully determined linear function of the input vector. Unless the input is truly noiseless *and* the true generative components can be exactly zero with finite probability *and* the learned basis exactly matches the true basis, the probability of any component being exactly zero is zero.

Our experiments with ICA of polyphonic piano spectra indicate that the activities of the resulting components are not in a simple correspondence with those of the underlying notes, and any transcription system based directly on ICA would require a more complex note decision process. In particular, the presence or absence of a particular note could not be assessed independently of other notes, because the activity of an ICA component could be "explained" by any of several notes. This would require the kind of "explaining away" effect which is an integral part of inference in multiple-cause latent variable models [60] of which the sparse coder is an instance.

The same considerations apply to independent subspace analysis (ISA, [20]), since it too is a noiseless model. On the other hand, the emergence of multiple dictionary elements with the same pitch suggests that the concept of independent subspaces is a relevant one. In the same way that the sparse coding system used here is a form of noisy ICA, the "sparse subspace coding"

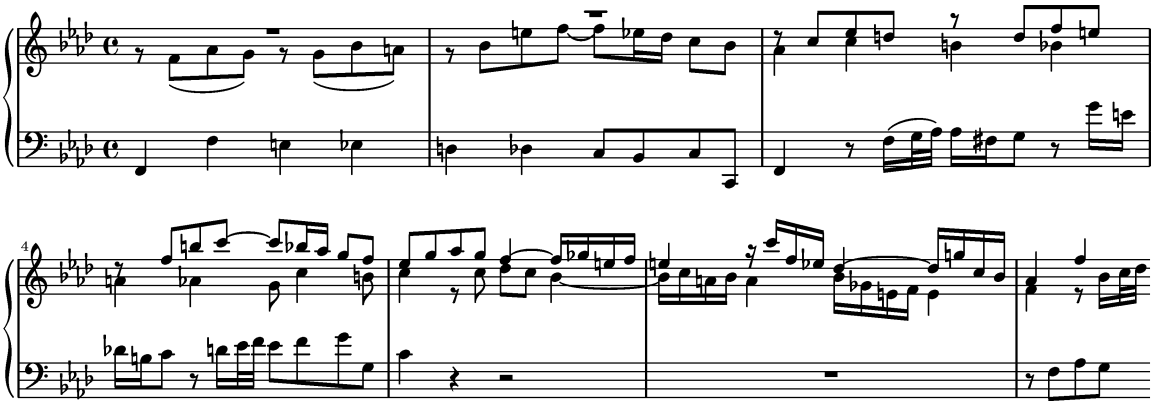

Fig. 14. Score of the opening of BWV795 (extracted from the complete score available from the Mutopia project, http://www.mutopiaproject.org).

discussed in Section V-A could be implemented using a generalization of ISA to include noise. This would enable the grouping of related components to be done automatically.

ISA has previously been applied to audio and music analysis [46], but the authors reported that dimensionality reduction of the input spectra was required to enable the basis vectors to converge to spectral features that were nonlocal in frequency. This is consistent with the fact that the piano dictionary is effectively rank-deficient, since noiseless models like ICA and ISA would be obliged to devote representational capacity to what is the null space of the sparse coding dictionary.

C. An Alternative Generative Model

It was noted in Section III-B that the generative model $\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{e}$ with Gaussian noise $\mathbf{e}$ is not accurate for short-term Fourier magnitude (or power) spectra. Under these circumstances the STFT power is at best a noisy estimate of the power spectral density (PSD) of the composite signal, with a standard deviation proportional to the power itself [61]. This mismatch between the model and the PSD makes itself felt when trying to fix an appropriate noise level for both quiet and loud passages, and over all regions of the spectrum. The problem is only partially alleviated by the normalization and gain control applied to the input; for example, referring to Figs. 13 and 14, the soft opening F is hardly detected, while, the high B in bar 4 (approximately 16.4 s into the piece) causes many components associated with unrelated notes to be activated.

One way to deal with this problem is to work with log-power spectra instead of power or magnitude spectra, which transforms multiplicative noise into additive noise, and is arguably a more appropriate representation for audio [27]. We note, however, that the resulting additive noise is still not Gaussian, but rather asymmetrically distributed with a heavy tail toward minus infinity [35], because of the divergent behavior of $\log u$ as $u \to -\infty$.

An alternative which we are beginning to investigate [35] is to model the multiplicative noise directly. We treat each of the m dictionary vectors as a discrete-frequency approximation to the PSDs of a Gaussian processes in the time domain. The PSD of the composite signal, $\mathbf{z} = \mathbf{A}\mathbf{s}$, is a weighted sum of the individual PSDs, while $\mathbf{x}$ will now denote the squared magnitude of the STFT, which we consider to be derived from $\mathbf{z}$ by a multiplicative noise process. Recognizing that the PSD of a Gaussian process is essentially a set of *variances*, and that the squared magnitude of the STFT consists of a set of small-sample estimates of those variances, we derive a specific form for the conditional density $p(\mathbf{x}|\mathbf{z})$ (a product of scaled Gamma distributions). This defines a new a energy function to replace (7) which can be optimized using a multiplicative algorithm based on those described in [33], [58] (see [35] for further details).

D. Parallels With Other Polyphonic Analysis Systems

We briefly mention two polyphonic transcription systems [4], [5] that can usefully be compared with the sparse coding approach. Both operate by sequentially extracting notes and subtracting their estimated spectra, leaving a residual to be analyzed at the next step. This can be likened to matching pursuit in the spectral domain and hence interpreted as a form of sparse decomposition, but unlike optimization-based methods such as ours, not one that maximizes any particular objective function. Another point of comparison is that the pitch-equivalence classes found in the piano dictionary are directly analogous to the multiple per-note spectral profiles used in [5]. It is interesting that the sparse coder was able to reproduce that particular processing strategy spontaneously through unsupervised learning, and suggests that the sparse coder be supplemented with an independent subspace analysis to find the pitch equivalence classes in an unsupervised way.

E. Alternative Inference Algorithms

The most computationally intensive aspect of the system described in this paper is the iterative optimization used to perform sparse decomposition, since each evaluation of the objective function (7) is of $O(nm)$ complexity, and, in our experiments, between 50 and 150 iterations were required per optimization to achieve convergence, depending on the sparsity of the result. This, in turn, depends on a number of factors including the model noise variance σ^2, the prior, and the extent to which the dictionary is matched to the data. Different performance characteristics could be obtained by using other algorithms for sparse decomposition, which include matching pursuit [62], basis pursuit (BP) and basis pursuit denoising (BPDN) [37], and the focal underdetermined system solver (FOCUSS)

algorithm [63]. Matching pursuit does not optimize any explicit objective, but BPDN and FOCUSS minimize functions of the form (7) for particular priors or "diversity costs." BP and BPDN use linear and quadratic programming respectively and assume an l_1 diversity cost, while FOCUSS uses an iterated reweighted least squares algorithm to minimize an l_α pseudo-norm for $\alpha \leq 1$. Dictionary learning algorithms based on FOCUSS have also been proposed [64]. Non-negative sparse decomposition [58] has been implemented using multiplicative optimization steps modeled on nonnegative matrix factorization [33].

VI. Conclusion

In this paper, we have applied a form of sparse coding based on a probabilistic multiple cause model to synthetically generated polyphonic harpsichord music and real polyphonic piano music. A model-based normalization procedure based on generalized exponential density fitting was developed (see (13) in Section III-A) to prepare the sequence of short-term Fourier spectra magnitudes (STFTMs) for presentation to the sparse coder, with the specific purpose equalizing the variance of the spectral "noise" at each frequency. The components of the STFTM were found to be very super-Gaussian, with most of the generalized exponential α_i parameters in the region of 0.5. Inference in the sparse coder was implemented using a novel optimization algorithm, while the dictionary learning algorithm was modified (the "decay when active correction") to improve its behavior when the inferred component activation is very sparse.

The results show that adaptive sparse coding can discover musically relevant structures in polyphonic mixtures, yielding accurate transcription in the case of the synthetic harpsichord and a highly structured dictionary consisting of groups of pitched spectra in the case of the piano.

However, the encodings produced in both experiments were generally not sparse enough to yield clean transcriptions without further processing to remove false-positives. This is partly due to the inaccurate additive noise model, which requires large peaks in the spectra, the relative amplitudes of which vary significantly even over the sounding of a single note, to be encoded as accurately as spectral regions of low activity, resulting in the need for several components to be active beyond those corresponding with the pitches present in the mixture. (The same comments apply doubly to an analysis based on ICA, since it assumes no noise at all.) Thus, the multiplicative noise model outlined in Section V-C is a logical development of the current system.

Finally, from the spontaneous emergence of spectral subspaces for each note, spanned by groups of dictionary vectors, we conclude that a form of "sparse subspace coding," derived from a noisy independent subspace model (using the multiplicative noise model mentioned above) could be a powerful framework within which to construct effective music analysis systems entirely through unsupervised learning.

Appendix

A. An Active-Set Quasi-Newton Optimizer

Inference in the sparse coder (Section II-B) involves solving the optimization problem (6) to find the vector $\mathbf{s}$ that maximizes the posterior $p(\mathbf{s}|\mathbf{x};\mathbf{A})$, or equivalently, minimizes the energy function $\mathcal{E}(\mathbf{s},\mathbf{x};\mathbf{A})$. While standard gradient based algorithms are effective when optimizing functions that are approximately quadratic near the optimum, the sparsity-inducing priors such as a Laplacian have a gradient discontinuity at zero, producing corresponding "creases" in the energy function wherever any component $s_i = 0$. If the minimum of the energy function occurs at one of these "creases," a gradient based optimizer will converge slowly, if at all, depending on the robustness of the implementation as regards step length heuristics and fallback procedures when the optimum is not locally quadratic.

To address this, a modified quasi-Newton optimizer was developed which explicitly deals with gradient discontinuities at component zeros of the vector being optimized, by maintaining a set of active dimensions. It could therefore be called an *active-set quasi-Newton* optimizer, though the same modification can be applied to any iterative gradient-based optimizer that involves a step length computation; for example, an active-set conjugate gradient version was also implemented.

Both the quasi-Newton optimizer and the conjugate gradient optimizer were based on algorithms found in [65] and the function FMINU in the MATLAB optimization toolbox. Details such as step length heuristics and termination conditions can be found in those sources and will not be stated here.

The optimization is an iterative procedure, involving a current point $\mathbf{s} \in \mathbb{R}^m$ and two sets I_0 and I_1 which contain the indices of the inactive and active components of $\mathbf{s}$, respectively, thus forming a partition of the set of indices: $I_0 \cap I_1 = \emptyset$, $I_0 \cup I_1 = \{1,\ldots,m\}$. Inactive coordinates are clamped to zero ($i \in I_0 \Longrightarrow s_i = 0$) and do not take part in the current optimization step, though they may subsequently be activated. To determine whether or not the ith coordinate should be active, a boolean indicator function $Q(\mathbf{s},i)$ is defined, which, assuming that $s_i = 0$, depends on the signs of the gradient of the cost function on either side of the point $s_i = 0$. These gradients are defined as

$$\partial_i^+ \mathcal{E}(\mathbf{s}) \stackrel{\text{def}}{=} \lim_{s_i \downarrow 0^+} \frac{\partial \mathcal{E}(\mathbf{s})}{\partial s_i} \quad \partial_i^- \mathcal{E}(\mathbf{s}) \stackrel{\text{def}}{=} \lim_{s_i \uparrow 0^-} \frac{\partial \mathcal{E}(\mathbf{s})}{\partial s_i} \tag{16}$$

where the limits are taken tending *down* to a value just above zero and *up* to a value just below zero respectively. The energy $\mathcal{E}(\mathbf{s},\mathbf{x};\mathbf{A})$ has been abbreviated to $\mathcal{E}(\mathbf{s})$ to reduce clutter. The indicator function is

$$Q(\mathbf{s},i) = \begin{cases} 0: & \operatorname{sgn} \partial_i^+ \mathcal{E}(\mathbf{s}) \geq 0 \wedge \operatorname{sgn} \partial_i^- \mathcal{E}(\mathbf{s}) \leq 0 \\ 1: & \text{otherwise} \end{cases} \tag{17}$$

in which $\operatorname{sgn} 0 \stackrel{\Delta}{=} 0$. If $Q(\mathbf{s},i) = 0$, then the point $\mathbf{s}$ represents a local minimum in the direction of the ith dimension, and so that dimension should be deactivated. Before the main optimization loop, we initialize by inactivating all coordinates, setting

$$I_0 = \{1,\ldots,m\} \quad I_1 = \emptyset \quad \mathbf{s} = \mathbf{0}.$$

Then, for each iteration of the main loop, the following steps are taken (see Fig. 15).

1) Compute proposed new point $\mathbf{s}'$ and step $\boldsymbol{\Delta}$ according to the base quasi-Newton or conjugate gradient algorithm, so that $\mathbf{s}' = \mathbf{s} + \boldsymbol{\Delta}$.

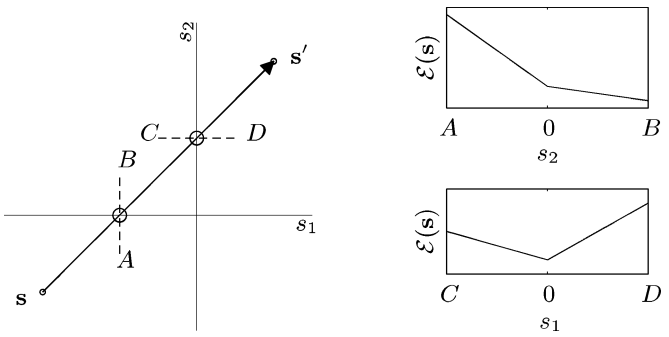

Fig. 15. A two-dimensional illustration of the operation of the modified active set optimizer. Given the proposed step from $\mathbf{s}$ to $\mathbf{s}'$, and the local behavior of the gradient along the segments AB and CD, the modified optimizer would truncate the step at the *second* zero crossing, and inactivate the *first* coordinate, s_1.

2) Check if the proposed step would result in any of the components of $\mathbf{s}$ changing sign, by defining the set of all such zero-crossings

$$Z = \{i \in I_1 | \operatorname{sgn} s_i \neq \operatorname{sgn} s_i'\}.$$

3) Check if any of the zero-crossings satisfy the inactivation criterion. First, define $\lambda(i)$ as the step size that leads to the zero crossing in the ith coordinate

$$\lambda(i) = -\frac{s_i}{\mathbf{\Delta}_i}$$

so that $[\mathbf{s} + \lambda(i)\mathbf{\Delta}]_i = 0$. Then define Z_0 as the set of zero crossings that satisfy the inactivation criterion

$$Z_0 = \{i \in Z | Q(\mathbf{s} + \lambda(i)\mathbf{\Delta}, i) = 0\}.$$

4) If there are any such zero crossings, truncate the step at the first zero crossing, otherwise, take the proposed step unmodified

$$\lambda^* = \begin{cases} \min_{i \in Z_0} \lambda(i): & Z_0 \neq \emptyset \\ 1: & Z_0 = \emptyset \end{cases}$$
$$\mathbf{s} \leftarrow \mathbf{s} + \lambda^* \mathbf{\Delta}.$$

5) Update the active and inactive sets to reflect the new current point $\mathbf{s}$. To do this, two sets are defined

$$I_- = \{i \in I_1 : Q(\mathbf{s}, i) = 0 \wedge s_i = 0\}$$
$$I_+ = \{i \in I_0 : Q(\mathbf{s}, i) = 1\}.$$

I_- is the set of currently active coordinates that should be deactivated. I_+ is the set of currently inactive coordinates eligible for reactivation. At this point, some heuristic decision must be made about which coordinates to activate, by choosing a set $I_+^* \subseteq I_+$. One option is to choose the coordinate which will result in the steepest descent when activated—this is the same criterion used in matching pursuit. Another is to choose a certain proportion of those eligible. Once the choice is made, the various elements are transferred between the active and inactive sets

$$I_0 \leftarrow (I_0 \setminus I_+^*) \cup I_-$$
$$I_1 \leftarrow (I_1 \setminus I_-) \cup I_+^*$$

where the operator $\setminus$ denotes set difference.

6) Perform any bookkeeping required by the base algorithm, e.g., update the inverse-Hessian approximation in a quasi-Newton algorithm. The sparsity of the current state can be used to make computational savings at this point, e.g., the inverse-Hessian will be a sparse matrix.

The main loop is subject to the same termination conditions as the unmodified optimizer, except that these apply to the currently active coordinates only. In addition, the main loop terminates if there are no active coordinates. By inactivating directions in which the local gradient is discontinuous, this algorithm keeps the Hessian approximation from becoming ill-conditioned, and if the activation is sparse, then only a small submatrix of the Hessian approximation is updated at each step.

A quantitative evaluation of the performance of the optimizer can be found in [30]; the main points are summarized here: The performance advantage of the modified optimizer becomes apparent only when significant numbers of units can be set to exactly zero. This happens if (a) the model noise variance $\sigma^2 = 1/\lambda$ is large, (b) the actual noise level in the data is low, (c) the prior is sharply peaked, and (d) the learned basis matrix $\mathbf{A}$ correctly matches the underlying sparse structure (if any) of the data. A standard quasi-Newton optimizer is faster for smooth priors, low model noise levels (i.e., large λ) and nonsparse activations. The quasi-Newton version was also compared with an equivalent active-set conjugate gradient algorithm, using the same termination conditions and the same line search procedure. In cases where a sparse decomposition was achieved, the quasi-Newton algorithm outperformed the conjugate gradient algorithm—the expense of maintaining the inverse Hessian approximation is mitigated by the sparsity of the updates.

B. "Decay When Active" Correction

The results of Section IV-A (see Fig. 7) show that, although the Lewicki-Sejnowski algorithm (12) is capable of managing the overall scaling (i.e., the Euclidean lengths) of the dictionary vectors without an explicit normalization step, the resulting scalings, and hence the scalings of the components s_j, is dependent on the sparsity of the true generative system. This can be explained by an analysis of the learning dynamics [30], which also shows that, if the sparsity of a component (measured as the probability of it being zero) is above a certain threshold, its corresponding dictionary vector will decay to zero. In order to counteract this effect, we derive here a correction to the algorithm which improves on the approximation required to derive (12) in the case that the sources s_j have a finite probability of being exactly zero; that is, the priors $p(s_j)$ are "mixed discrete/continuous."

The update rule (12) includes a "decay" term, which causes each basis vector to decay at a rate proportional to its length. The modification is simply this: We only decay those columns of $\mathbf{A}$ whose corresponding components are actually active in the MAP estimate $\hat{\mathbf{s}}$, i.e., $s_j \neq 0$. This can be justified as follows. Assume the prior for the jth component is

$$p(s_j) = \zeta_j p_*(s_j) + (1 - \zeta_j)\delta(s_j) \tag{18}$$

where ζ_j is the probability that s_j is not zero, p_* is the density function for an active component, and δ is the Dirac delta distribution. This can be modeled by introducing a set of binary latent variables u_j

$$p(s_j) = \sum_{u_j \in \{0,1\}} p(s_j|u_j)p(u_j) \tag{19}$$

where the prior and conditional probabilities are defined as

$$\begin{aligned} p(s_j|u_j = 0) &= \delta(s_j) \quad p(u_j = 0) = 1 - \zeta_j \\ p(s_j|u_j = 1) &= p_*(s_j) \quad p(u_j = 1) = \zeta_j. \end{aligned} \tag{20}$$

If we introduce the binary tuple $\mathbf{u} = (u_1, \ldots, u_m)$, and define the sets $I_1(\mathbf{u}) = \{j|u_j = 1\}$ and $I_0(\mathbf{u}) = \{j|u_j = 0\}$ as the sets of active and inactive components respectively, then the joint density $p(\mathbf{x}, \mathbf{s}; \mathbf{A})$ can be written as

$$\begin{aligned} p(\mathbf{x}, \mathbf{s}; \mathbf{A}) = \sum_{\mathbf{u} \in \{0,1\}^m} p(\mathbf{x}|\mathbf{s}; \mathbf{A}) \prod_{j \in I_0(\mathbf{u})} (1 - \zeta_j)\delta(s_j) \\ \times \prod_{j' \in I_1(\mathbf{u})} \zeta_{j'} p_*(s_{j'}). \end{aligned} \tag{21}$$

Next, we assume that this sum is dominated by a single term in $\hat{\mathbf{u}}$, the MAP estimate which is defined to have a pattern of activity derived from $\hat{\mathbf{s}}$, the MAP estimate of $\mathbf{s}$ obtained, not by using the mixture prior, but by using the more easily optimized prior $p_*(\cdot)$. The components of $\hat{\mathbf{u}}$ are

$$\hat{u}_j = \begin{cases} 0: & \hat{s}_j = 0 \\ 1: & \hat{s}_j \neq 0. \end{cases} \tag{22}$$

The overall dependence on $\mathbf{s}$ of the single-term approximation to the posterior is

$$p(\mathbf{s}|\mathbf{x}; \mathbf{A}) \propto p(\mathbf{x}|\mathbf{s}; \mathbf{A}) \prod_{j \in I_1(\mathbf{u})} p_*(s_j) \prod_{j' \in I_0(\mathbf{u})} \delta(s_{j'}). \tag{23}$$

The derivation now follows that of [22], making a saddle-point approximation to the posterior at $\hat{\mathbf{s}}$, but in this case, the density is confined to the dimensions $j \in I_1(\hat{\mathbf{u}})$, with $s_j = 0$ for all $j \in I_0(\hat{\mathbf{u}})$. Thus, its covariance matrix $\mathrm{cov}[\mathbf{s}]$ is zero outside the square submatrix defined by the active components $I_1(\hat{\mathbf{u}})$. The results will be simpler to write if we permute the indices so that the active components come before the inactive ones, so that, e.g., $\mathbf{A} \equiv (\overline{\mathbf{A}}\ \underline{\mathbf{A}})$, where $\overline{\mathbf{A}}$ and $\underline{\mathbf{A}}$ are matrices containing the active and inactive dictionary vectors respectively. The inner expectation in (10), required to compute the derivative of the log-likelihood $\mathcal{L}(\mathbf{A})$, can then be written as

$$\begin{aligned} \langle \mathbf{e}\mathbf{s}^T \rangle_{\mathbf{s}|\mathbf{x};\mathbf{A}} &= \hat{\mathbf{e}}\hat{\mathbf{s}}^T - \mathbf{A}\mathrm{cov}[\mathbf{s}] \\ &= \hat{\mathbf{e}}(\bar{\hat{\mathbf{s}}}^T \ \ \mathbf{0}) - (\overline{\mathbf{A}}\ \ \underline{\mathbf{A}}) \begin{pmatrix} \overline{\mathbf{H}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\ &= \left([\hat{\mathbf{e}}\bar{\hat{\mathbf{s}}}^T - \overline{\mathbf{A}}\overline{\mathbf{H}}^{-1}] \quad \mathbf{0} \right) \end{aligned} \tag{24}$$

where $\bar{\hat{\mathbf{s}}}$ is the vector of active components, $\overline{\mathbf{H}}$ is the Hessian of the posterior evaluated for the active coordinates only, and $\mathbf{0}$ is a zero matrix of the appropriate size. If this is used as a stochastic gradient update step, then only the active columns of $\mathbf{A}$ will be modified; there is no decay term for the inactive columns.

Turning to the alternative version of the update rule (12), a simple equivalent can be obtained by premultiplying (24) by $\overline{\mathbf{A}}\ \overline{\mathbf{A}}^T$ instead of the $\mathbf{A}\mathbf{A}^T$ used in [22]. Making the further approximation $\overline{\mathbf{H}} \approx \lambda \overline{\mathbf{A}}^T \overline{\mathbf{A}}$ [22], and noting that $\lambda \overline{\mathbf{A}}^T \hat{\mathbf{e}} = \gamma(\bar{\hat{\mathbf{s}}})$ at an extremum of the energy $\mathcal{E}(\mathbf{s}, \mathbf{x}; \mathbf{A})$, we obtain

$$\begin{aligned} \lambda \overline{\mathbf{A}}\overline{\mathbf{A}}^T \langle \mathbf{e}\mathbf{s}^T \rangle &= \lambda \overline{\mathbf{A}} \left([\overline{\mathbf{A}}^T \hat{\mathbf{e}} \bar{\hat{\mathbf{s}}}^T - \overline{\mathbf{A}}^T \overline{\mathbf{A}}\overline{\mathbf{H}}^{-1}] \quad \mathbf{0} \right) \\ &= \overline{\mathbf{A}} \left(\left[\gamma(\bar{\hat{\mathbf{s}}}) \bar{\hat{\mathbf{s}}}^T - \mathbf{I} \right] \quad \mathbf{0} \right). \end{aligned} \tag{25}$$

The final update rule for the active columns of $\mathbf{A}$ is therefore

$$\overline{\mathbf{A}} \leftarrow \overline{\mathbf{A}} + \eta \overline{\mathbf{A}} \left[\gamma(\bar{\hat{\mathbf{s}}}) \bar{\hat{\mathbf{s}}}^T - \mathbf{I} \right] \tag{26}$$

with no change to the inactive columns.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their many constructive comments and suggestions.

REFERENCES

[1] J. A. Moorer, "On the Segmentation and Analysis of Continuous Musical Sound," Ph.D. dissertation, CCRMA, Stanford Univ., Stanford, CA, 1975.

[2] J. P. Bello-Correa, "Toward the Automated Analysis of Simple Polyphonic Music: a Knowledge-Based Approach," Ph.D. dissertation, Dept. Elect. Eng., Queen Mary, Univ. London, London, U.K., 2003.

[3] A. Sterian, "Model Based Segmentation of Time-Frequency Images for Musical Transcription," Ph.D. dissertation, Dept. Elect. Eng., Univ. Michigan , Ann Arbor, MI, 1999.

[4] A. Klapuri, T. Virtanen, and J.-M. Holm, "Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals," in *Proc. COST-G6 Conf. Digital Audio Effects*, Verona, Italy, 2000, DAFx-00, pp. 141–146.

[5] P. Lepain, "Polyphonic pitch extraction from musical signals," *J. New Music Res.*, vol. 28, no. 4, pp. 296–309, 1999.

[6] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Trans. Multimedia*, vol. 6, no. 3, pp. 439–449, Jun. 2004.

[7] L. Rossi, G. Girolami, and L. Leca, "Identification of polyphonic piano signals," *Acustica*, vol. 83, no. 6, pp. 1077–1084, 1997.

[8] K. D. Martin, "A Blackboard System for Automatic Transcription of Simple Polyphonic Music," M.I.T. Media Lab, Perceptual Computing, Tech. Rep. 385, 1996.

[9] K. Kashino, K. Nakadai, T. Kinoshita, and H. Tanaka, "Application of the Bayesian probability network to music scene analysis," in *Computational Auditory Scene Analysis*, D. F. Rosenthal and H. Okuno, Eds. Mahwah, NJ: Lawrence Erlbaum, 1998, pp. 115–137.

[10] A. Khurshid and S. L. Denham, "A temporal-analysis-based pitch estimation system for noisy speech with a comparative study of performance of recent systems," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1112–1124, Sep. 2004.

[11] E. Cambouropoulos, "Toward a General Computational Theory of Musical Structure," Ph.D. dissertation, Faculty Music, Dept. Artif. Intell., Univ. Edinburgh, 1998.

[12] F. Attneave, "Some informational aspects of visual perception," *Pyschol. Rev.*, vol. 61, no. 3, pp. 183–193, 1954.

[13] H. B. Barlow, "Possible principles underlying the transformation of sensory messages," in *Sensory Communication*, W. A. Rosenblith, Ed. Cambridge, MA: MIT Press, 1961, pp. 217–234.

[14] E. P. Simoncelli, "Vision and the statistics of the visual environment," *Current Opinion Neurobiol.*, vol. 13, pp. 144–149, 2003.

[15] H. B. Barlow, "Redundancy reduction revisited," *Network: Computation in Neural Systems*, vol. 12, no. 3, pp. 241–253, 2001.

[16] ——, "Banishing the homonculus," in *Perception as Bayesian Inference*, D. C. Knill and W. Richards, Eds. Cambridge, UK: Cambridge University Press, 1996, p. 425.
[17] R. P. N. Rao and B. A. O. M. S. Lewicki, *Probabilistic Models of the Brain: Perception and Neural Function*. Cambridge, MA: MIT Press, 2000.
[18] D. J. Field and B. A. Olshausen, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.
[19] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proc. Royal Soc. London B*, vol. 265, pp. 2315–2320, 1998.
[20] A. Hyvärinen and P. Hoyer, "Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces," *Neural Comput.*, vol. 12, no. 7, pp. 1705–1720, 2000.
[21] B. A. Olshausen, "Learning Linear, Sparse, Factorial Codes," Artificial Intelligence Lab, MIT, Tech. Rep. AI Memo 1580, 1996.
[22] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Comput.*, vol. 12, pp. 337–365, 2000.
[23] K. Kreutz-Delgado, B. D. Rao, K. Engan, T.-W. Lee, and T. Sejnowski, "Convex/Schur-convex (CSC) log-priors and sparse coding," in *Proc. 6th Joint Symp. Neural Computation*, Pasadena, CA, May 1999.
[24] P. Walmsley, "Signal Separation of Musical Instruments," Ph.D. dissertation, Dept. Eng., Cambridge Univ., Cambridge, U.K., 1998.
[25] C. Raphael, "Automatic transcription of piano music," in *Proc. 3rd Intl. Symp. Music Information Retrieval (ISMIR 2002)*, Paris, France, 2002, pp. 15–19.
[26] A. T. Cemgil, B. Kappen, and D. Barber, "Generative model based polyphonic music transcription," in *2003 IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2003, pp. 181–184.
[27] E. Vincent and X. Rodet, "Music transcription with isa and hmm," in *Proc. 5th Symp. Independent Component Analysis and Blind Source Separation (ICA 2004)*, Granada, Spain, 2004, to be published.
[28] M. Casey, "Auditory Group Theory With Applications to Statistical Basis Methods for Structured Audio," Ph.D. dissertation, MIT Media Lab., Cambridge, MA, 1998.
[29] P. Smaragdis, "Redundancy Reduction for Computational Audition, a Unifying Approach," Ph.D. dissertation, MIT Media Lab., Cambridge, MA, 2001.
[30] S. A. Abdallah, "Toward Music Perception by Redundancy Reduction and Unsupervised Learning in Probabilistic Models," Ph.D. dissertation, Dept. Electron. Eng., King's College London, London, U.K., 2002.
[31] M. S. Lewicki, "Efficient coding of natural sounds," *Nature Neuroscience*, vol. 5, no. 4, pp. 356–363, 2002.
[32] P. Smaragdis, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 2003, pp. 177–180.
[33] D. D. Lee and H. S. Seung, "Algorithms for nonnegative matrix factorization," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 556–562.
[34] M. D. Plumbley, "Algorithms for nonnegative independent component analysis," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 534–543, May 2003.
[35] S. A. Abdallah and M. D. Plumbley, "Polyphonic transcription by nonnegative sparse coding of power spectra," in *Proc. 5th Int. Symp. Music Information Retrieval (ISMIR 2004)*, Barcelona, Spain, 2004.
[36] G. F. Harpur, "Low Entropy Coding With Unsupervised Neural Networks," Ph.D. dissertation, Dept. Eng., Cambridge Univ., Cambridge, U.K., 1997.
[37] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.
[38] A. Hyvärinen, "Sparse code shrinkage: denoising of non-Gaussian data by maximum-likelihood estimation," *Neural Comput.*, vol. 11, no. 7, pp. 1739–1768, 1999.
[39] M. S. Lewicki and B. A. Olshausen, "A probabilistic framework for the adaptation and comparison of image codes," *J. Opt. Soc. Amer. A: Opt. Image Sci. Vision*, 1999.
[40] D. C. Knill and W. Richards, Eds., *Perception as Bayesian Inference*. Cambridge, U.K.: Cambridge University Press, 1996.
[41] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "Independent component analysis and (simultaneous) third-order tensor diagonalization," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2262–2271, Oct. 2001.
[42] S. A. Abdallah and M. D. Plumbley, "If edges are the independent components of natural scenes, what are the independent components of natural sounds?," in *Proc. 3rd Intl. Conf. Independent Component Analysis and Signal Separation, ICA2001*, San Diego, CA, 2001, pp. 534–539.
[43] T. Blumensath and M. Davies, "Unsupervised learning of sparse and shift-invariant decompositions of polyphonic music," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP 2004)*, vol. 5, Montreal, Canada, May 2004, pp. 497–500.
[44] H. Helmholtz, *On the Sensations of Tone*. New York: Dover, 1954/1885.
[45] O. Schwartz and E. P. Simoncelli, "Natural sound statistics and divisive normalization in the auditory system," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds: MIT Press, 2001, vol. 13, pp. 166–172.
[46] M. A. Casey and A. Westner, "Separation of mixed audio sources by independent subspace analysis," in *Proc. Int. Computer Music Conf. (ICMC 2000)* , Berlin, Germany, 2000, pp. 154–161.
[47] *The Principal Component Structure of Natural Sound*, L. G. Iordanov and P. S. Penev. (1999). *http://citeseer.nj.nec.com/iordanov99principal.html* [Online]
[48] R. Everson and S. Roberts, "Independent component analysis: a flexible nonlinearity and decorrelating manifold approach," *Neural Comput.*, vol. 11, no. 8, pp. 1957–1983, 1999.
[49] L. Zhang, A. Cichocki, and S. ichi Amari, "Self-adaptive blind source separation based on activation functions adaptation," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 233–244, Mar. 2004.
[50] E. J. Anderson, "Limitations of Short-Time Fourier Transforms in Polyphonic Pitch Recognition," Dept. Comp. Sci. Eng., Univ. Washington, 1997.
[51] A. Klapuri, "Multipitch estimation and sound separation by the spectral smoothness principle," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing, ICASSP 2001*, vol. 5, Salt Lake City, Utah, 2001, pp. 3381–3384.
[52] S. A. Abdallah and M. D. Plumbley, "Unsupervised onset detection: a probabilistic approach using ICA and a hidden Markov classifier," in *Cambridge Music Processing Colloquium*, Cambridge, UK, 2003.
[53] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in music signals," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 1035–1047, Sep. 2005.
[54] B. A. Olshausen and K. J. Millman, "Learning sparse codes with a mixture-of-Gaussians prior," in *Advances in Neural Information Processing Systems*, T. K. Leen and K.-R. Müller, Eds: MIT Press, 2000, vol. 12, pp. 841–847.
[55] R. W. Young, "Inharmonicity of plain wire piano strings," *J. Acoust. Soc. Amer.*, vol. 24, no. 3, pp. 267–272, 1952.
[56] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 3017–3030, Dec. 1996.
[57] A. S. Bregman, *Auditory Scene Analysis*. Cambridge, MA: MIT Press, 1990.
[58] P. O. Hoyer, "Non-negative sparse coding," in *Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing)*, Martigny, Switzerland, 2002, pp. 557–565.
[59] E. D. Scheirer and B. L. Vercoe, "SAOL: the MPEG-4 structured audio orchestra language," *Comput. Music J.*, vol. 23, no. 2, pp. 31–51, 1999.
[60] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan-Kaufmann, 1988.
[61] S. M. Kay, *Modern Spectral Estimation*. Englewood Cliffs, NJ: Prentice Hall, 1988.
[62] S. G. Mallat and Z. Zang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.
[63] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, Mar. 1997.
[64] K. Kreutz-Delgado and B. D. Rao, "FOCUSS-based dictionary learning algorithms," in *Proc. SPIE Volume 4119: Wavelet Applications in Signal and Image Processing VIII*, A. Aldroubi, A. F. Laine, and M. A. Unser, Eds., Bellingham, Washington, 2000, pp. 459–473.
[65] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, Eds., *Numerical Recipes in C*. Cambridge, UK: Cambridge University Press, 1992.

Samer A. Abdallah received the B.A. degree in natural sciences from Cambridge University, Cambridge, U.K., in 1994, after which he spent three years working in industry. He then received the M.Sc. and Ph.D. degrees from King's College London, London, U.K., in 1998 and 2003, respectively.

He is now a postdoctoral Researcher at the Centre for Digital Music, Queen Mary, University of London. His research interests include music perception, unsupervised learning and Bayesian networks.

Mark D. Plumbley (S'88–M'90) began research on neural networks in 1987, as a Research Student under the late Frank Fallside at the Engineering Department, Cambridge University, continuing as a Research Associate using genetic algorithms to modify neural networks.

He joined the Centre for Neural Networks at King's College London in 1991, moving to the Department of Electronic Engineering in 1995. In 2002, he joined the new DSP \& Multimedia Group at Queen Mary, University of London, and is currently working on independent component analysis (ICA), with particular interest on applications to the analysis and separation of audio and music signals.