Deformable Spectrograms

Manuel Reyes-Gomez, Nebojsa Jojic, Daniel P.W. Ellis,

Abstract

Speech and other natural sounds show high temporal correlation and smooth spectral evolution punctuated by a few, irregular and abrupt changes. In a conventional Hidden Markov Model (HMM), such structure is represented weakly and indirectly through transitions between explicit states representing 'steps' along such smooth changes. It would be more efficient and informative to model successive spectra as *transformations* of their immediate predecessors, capturing the evolution of the signal energy through time. We present a model which focuses on local deformations of adjacent bins in a time-frequency surface to explain an observed sound, using explicit representation only for those bins that cannot be predicted from their context. We further decompose the log-spectrum into two additive layers, which are able to separately explain and model the evolution of the harmonic excitation, and formant filtering of speech and similar sounds. Smooth deformations are modeled with hidden transformation variables in both layers, using Markov Random fields (MRFs) with overlapping subwindows as observations; inference is efficiently performed via loopy belief propagation. The model can fill-in deleted time-frequency cells without any signal model, and an entire signal can be compactly represented with a few specific states along with the deformation maps for both layers. We present results on a speech recognition task, that suggest that the model discovers a global structure on the dynamics of the signal's energy that helps to alleviate the problems generated by noise interferences.

Index Terms

signal models, spectrogram, pitch and formants dynamics, belief propagation

I. Introduction

Hidden Markov Models (HMMs) work best when only a limited set of distinct states need to be modeled, as in the case of speech recognition where the models need only be able to discriminate

M. Reyes-Gomez is now with Microsoft Research, work developed both at the Department of Electrical Engineering, Columbia University and Microsoft Research; N. Jojic is with Microsoft Research; D. Ellis is with the Department of Electrical Engineering, Columbia University.

between phone classes. When HMMs are used with the express purpose of accurately modeling the full detail of a rich signal such as speech, they require a large number of states. In [1], HMMs with 8,000 states were required to accurately represent one person's speech for a source separation task. The large state space is required because it attempts to capture every possible instance of the signal. If the state space is not large enough, the HMM will not be a good generative model since it will end up with a "blurry" set of states which represent an average of the features of different segments of the signal, and cannot be used in turn to "generate" the signal.

In many audio signals including speech and musical instruments, there is a high correlation between adjacent frames of their spectral representation. Our approach consists of exploiting this correlation so that explicit models are required only for those frames that cannot be accurately predicted from their context. In [2], context is used to increase the modeling power of HMMs, while keeping a reasonable size of parameter space, however the correlation between adjacent frames is not explicity modeled. Our model captures the general properties of such audio sources by modeling the evolution of their harmonic components. Based on the widely-used source-filter model for such signals, we devise a layered generative graphical model that describes these two components in separate layers: one for the excitation harmonics, and another for resonances such as vocal tract formants. This layered approach draws on successful applications in computer vision that use layers to account for different sources of variability [3], [4], [5]. Our approach explicitly models the self-similarity and dynamics of each layer by fitting the log-spectral representation of the signal in frame t with a set of transformations of the log-spectra in frame $t-1$. As a result, we do not require separate states for every possible spectral configuration, but only a limited set of "sharp" (not blurry) states that can still cover the full spectral variety of a source via such transformations. This approach is thus suitable for any time series data with high correlation between adjacent observations.

We will first quickly review of the graphical model framework including the belief propagation algorithm and some of the terminology used in the paper. Then we will introduce a model that captures the spectral deformation field of the speech harmonics, and show how this can be exploited to interpolate missing observations. Following that, we introduce the two-layer model that separately models the deformation fields for harmonic and formant resonance components, and show that such a separation is necessary to describe speech signals accurately through examples of interpolating missing data based on one and two layers. Then we will present the complete model including the two deformation fields and the "sharp" states. Using only a few states in combination with the two deformation fields, this model can accurately reconstruct the signal. We present results on a speech recognition task that suggest that the model discovers a global structure in the dynamics of the signal's energy that helps to alleviate the problems generated by noisy interference. Finally we discuss a modification of this model to segment a mixture of speakers into dominant speaker regions, an application to be described in more detail in a forthcoming paper.

Early developments of this work were presented in [6] and [7]. In this paper we present a longer version with more complete implementation details and results on a speech recognition task not presented in the previous publications.

II. Probabilistic Graphical Models

The graphical models framework is an intuitive and modular way to model complex systems as a graphical structure of simpler parts. The observed variables of the system as well as the unknown or hidden variables are represented using nodes. Observed variables which have known fixed values are represented by shaded nodes, while hidden variables, modeled as random variables, are illustrated by unshaded nodes. Sets of variables that have direct interaction with each other are connected through edges, forming a graphical representation of the system. Probability theory permits us to investigate or to query the state of the unknown variables given the observed variables, a process known as inference. Graphical models are divided into two major classes: directed and undirected graphical models.

A. Directed Graphical Models

In directed graphical models, the edges between variables have a notion of causality and therefore are represented by edges with directions or arrows. The set of nodes (X_{π_i}) that have arrows pointing into node X_i are referred as the parents of X_i. In directed graphs, the causal relationship between a node and its parents is defined by conditional probabilities p($X_i \mid X_{\pi_i}$). The joint probability p(X_1,X_2,X_3,...,X_n) between all variables (hidden and observed) in the system is defined as:

$$p(X_1, X_2, X_3, ..., X_n) = \prod_{i=1}^{n} p(X_i \mid X_{\pi_i}). \tag{1}$$

A directed graphical model widely used to model speech and audio is the hidden Markov model (HMM), (figure 1a). There, hidden nodes X = [X_0,X_1,..,X_T] represent the acoustic class at frame t, while the observed variables Y = [Y_0,Y_1,...,Y_T] represent features of the audio signal. The conditional probabilities in this model are defined by p($X_{t+1} \mid X_t$) and p($Y_t \mid X_t$). The conditional probabilities p($X_{t+1} = j \mid X_t = i$) = $\pi_{(i,j)}$ =$\pi_{(X_t,X_{t+1})}$, are represented as entries on an $N \times N$ transition probabilities

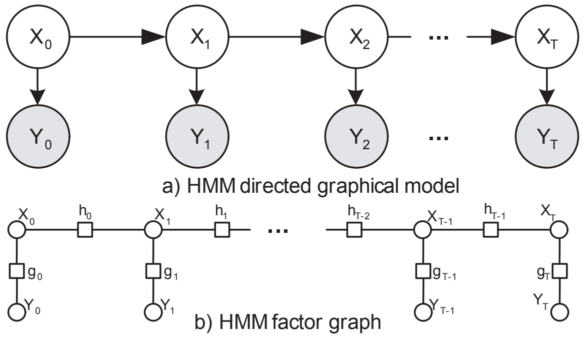

Fig. 1. a) An HMM as a directed graphical model, b) HMM factor graph representation.

matrix π, where N is the number of different acoustic classes that X_t can take. The local likelihood conditional probabilities p$(Y_t \mid X_t)$ are frequently modeled with a Gaussian distribution (or a mixture of Gaussians) such that p$(Y_t \mid X_t = i)$ = $\mathcal{N}(Y_t; \mu_i, \Sigma_i)$. Then the N acoustic classes are defined by N sets of parameters θ_i = (μ_i, Σ_i). The parameters θ that define an HMM are $\theta = \{\pi, \mu_1, \Sigma_1, \mu_2, \Sigma_2, ..., \mu_N, \Sigma_N\}$.

For a given model with parameters θ and observations Y, we would like to find the best set of parameters that maximize the likelihood of the observations given the model, a process known as "maximum likelihood parameters estimation". The expectation-maximization (EM) algorithm provides a general approach to the problem of maximum likelihood parameter estimation in statistical graphical models. In this approach the log-likelihood of the model p$(Y \mid \theta)$ is lower bounded by a auxiliary function, $\mathcal{L}(q, \theta)$, defined as:

$$\log p(Y \mid \theta) \geq \mathcal{L}(q, \theta) = \sum_X q(X \mid Y) \log \frac{p(X, Y \mid \theta)}{q(X \mid Y)} \tag{2}$$

where the term $q(X \mid Y)$ is regarded as the averaging function approximating the posterior (see below). The EM algorithm is essentially a coordinate ascent algorithm on the auxiliary function $\mathcal{L}(q, \theta)$. In the $t + 1^{th}$ iteration, q_{t+1} is found as the choice of q that maximizes $\mathcal{L}(q, \theta_t)$ given the current set of parameters θ_t. Then q_{t+1} is used to maximize $\mathcal{L}(q_{t+1}, \theta)$ with respect to θ to find θ_{t+1}. Further iterations of the algorithm are made to find q_{t+2}, θ_{t+2}, etc. The above steps give the algorithm its name since they are regarded as:

$$\textbf{E}\text{xpectation Step} \qquad argmax_q(\mathcal{L}(q, \theta_t)) \tag{3}$$

$$\textbf{M}\text{aximization Step} \qquad argmax_\theta(\mathcal{L}(q_{t+1}, \theta)) \tag{4}$$

It can be shown [8] that the choice for the averaging function $q_{t+1}(X \mid Y)$ that maximizes $\mathcal{L}(q, \theta_t)$ on the E step is $p(X \mid Y, \theta_t)$, the posterior probability of the hidden variables given the observations and the latest estimated parameters. The M step is done by taking the derivatives of $\mathcal{L}(q_{t+1}, \theta)$ with respect to of each one of the parameters in θ and solving the equations. The E step is also referred to as the inference procedure since it consists of inferring the state of the model's hidden variables, by computing their joint probability given the observed variables and the model parameters. The M step is also referred to as "learning" since it is the process of estimating the best set of parameters for the model given the observations. The posterior probability of the hidden variables given the observation can be found using Bayes theorem:

$$p(X \mid Y, \theta_t) = \frac{p(X, Y \mid \theta_t)}{p(Y \mid \theta_t)} \tag{5}$$

The numerator of eqn. 5 for the case of an HMM is defined using eqn. 1 as:

$$p(X, Y \mid \theta_t) = \prod_{t=0}^{T-1} p(X_{t+1} \mid X_t) \prod_{t=0}^{T} p(Y_t \mid X_t) \tag{6}$$

The overall likelihood of the model, $P(Y \mid \theta)$, can be obtained by marginalizing 6 with respect to X, resulting in:

$$p(Y \mid \theta) = \sum_{X_0} \sum_{X_1} \cdots \sum_{X_T} \prod_{t=0}^{T-1} p(X_{t+1} \mid X_t) \prod_{t=0}^{T} p(Y_t \mid X_t) \tag{7}$$

At first sight it seems that we need to perform N^T summations since we have T variables X_t with N values each. However, the factorized form of the joint probability distribution (eqn. 6) permits us to organize the summations by moving the relevant factors inside as shown in eqn. 8, reducing the total number of summations needed and revealing useful recursions.

$$p(Y) = \sum_{X_T} \cdots \sum_{X_1} p(X_2 \mid X_1) p(Y_1 \mid X_1) \sum_{X_0} p(X_1 \mid X_0) p(Y_0 \mid X_0) \tag{8}$$

The sum-product algorithm (described later) systematically exploits the factorization of the joint probability distribution to perform exact inference in complex graphical models.

B. Undirected Graphical Models

Undirected graphical models, also known as Markov random fields (MRFs) lack the notion of causality that the directed models have, eliminating the use of the directions on the edges. They are used in systems where local constraints between connected nodes can be expressed, but where it is hard to ensure that the conditional probabilities at different nodes are consistent with each other. Local parameterization was done in directed graphs through the use of conditional probabilities; in undirected graphs such parameterization is done through the use of *potential functions*, which are structured to favor certain local configurations of variables by assigning them a larger value. They are assumed to be strictly positive, real-valued functions, but are otherwise arbitrary. In general, potential functions are neither conditional probabilities nor marginal probabilities and in this sense they do not have a local probabilistic interpretation. The product of the potential functions is, however, still required to represent the joint distribution of all the variables, hidden and observed, in the graphical model:

$$p(X) = \frac{1}{Z} \prod_{S} \psi_{X_S}(X_S) \tag{9}$$

where ψ_{X_S} represents the potential function defined on the subset of variables X_S, and Z is a normalization constant. Eqn. 9 permits the likelihood of the model to be factorized in simpler terms allowing a tractable way to perform the inference of the model. Exact inference, however, is not always possible for either type of models. This can occur when the conditional distributions or the potential functions involve a large number of variables, reducing the model factorization capabilities, or if the model has some specific topological characteristic that will be discussed later. Exact inference, when possible, for both types of models can be achieved through the use of several similar algorithms: the junction tree algorithm, Pearl's propagation algorithm, and the sum-product algorithm. In this paper we relied in the sum-product algorithm since it is easily extended to perform approximate inference on intractable models. The sum-product algorithm is defined in terms of the *factor graph* representation of a graphical model.

C. Factor Graphs

Factor graphs have been explicitly designed to work with algorithms that exploit the factorization of a complex function p with domain X into simpler functions ψ_{X_S} defined over subsets X_S of set X. Just as in eqn. 9, or in eqn. 1 with $\psi_{X_S} = p(X_i \mid X_{\pi_i})$ and $X_S = \{X_i, X_{\pi_i}\}$. **Definition:** A factor graph is a bipartite graph that expresses the structure of a factorization such as eqn. 9. A factor graph has a variable node for each variable X_i, a factor node for each local function ψ_{X_S}, and an edge connecting variable node X_i to factor node ψ_{X_S} if only and only if X_i is an argument of ψ_{X_S}, i.e. $X_i \in X_S$ [9]. Variable nodes are represented with circles, while function nodes are represented with squares. Figure 1b shows the factor graph for an HMM. In the figure the notation of the function nodes is equivalent to: $g_k = p(Y_k \mid X_k)$ and $h_k = p(X_{k+1} \mid X_k)$.

D. Sum-Product Algorithm

Coming back to the likelihood $P(Y \mid \theta)$ of an HMM (eqn. 8), notice that the right-most summation, $\sum_{X_0} p(X_1 \mid X_0)p(Y_0 \mid X_0)$ can be seen as a function $f(X_1)$ of variable X_1. The second-rightmost summation can be expressed as: $\sum_{X_1} p(X_2 \mid X_1)p(Y_1, X_1)f(X_1$, which is a function $f(X_2)$ of variable

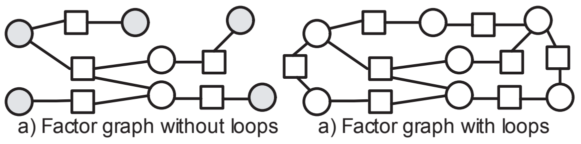

Fig. 2. Shaded nodes represent leaf nodes

X_2 and so on. Each one of the summations is marginalizing one of the variables in the model. The sum-product algorithm is an efficient procedure for computing marginal functions that exploits the factorization of the global function, using the distributive law to simplify the summations and reuse intermediate partial sums. The "flow" of intermediate products and summations used by the algorithm is conceptualized as a set of messages between the nodes of the factor graph representation of the model. The update rules for those messages are defined as:

Message from variable x to local function f:

$$m_{x \to f}(x) = \prod_{h \in g_x \setminus f} m_{h \to x}(x) \tag{10}$$

where g_x represents all the functions that have x as one of its arguments. The messages consist of all the incoming messages into node x, except the one coming from node f.

Message from local function to variable:

$$m_{f \to x}(x) = \sum_{\sim x} f(X) \prod_{y \in n(f) \setminus x} m_{y \to f}(y) \tag{11}$$

where $X = n(f)$ is the set of arguments of the function f, and $\sum_{\sim x}$ represents the summations of all the arguments in X excepting x. Notice that both kind of messages are functions of variable x. Variable-to-function messages can be interpreted as the "belief" that the variable has, of itself, given to the values of all its other functions. Function-to-variable messages can be interpreted as the "belief" that the function has with respect to the variable's state, given the states of all the other variables in the function's argument.

The algorithm starts sending messages through the leaf nodes (fig. 2a). Here, only nodes representing hidden variables are considered. Observed variable nodes just send identity messages. *Condition one*.- A node sends a message once all the incoming messages needed to send that message have been received.

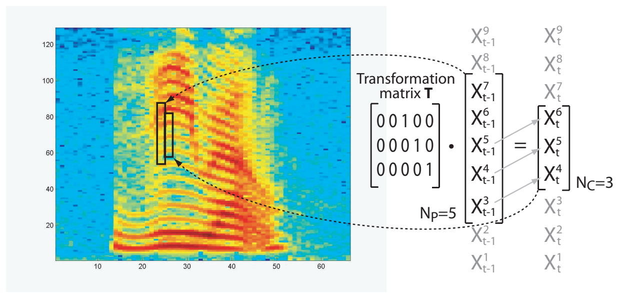
Transformation matrix **T**
$\begin{bmatrix} 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{bmatrix}$
$N_P=5$
$N_C=3$

Fig. 3. The $N_C = 3$ patch of time-frequency bins outlined in the spectrogram can be seen as an “upward” version of the marked $N_P = 5$ patch in the previous frame. This relationship can be described using the matrix shown.

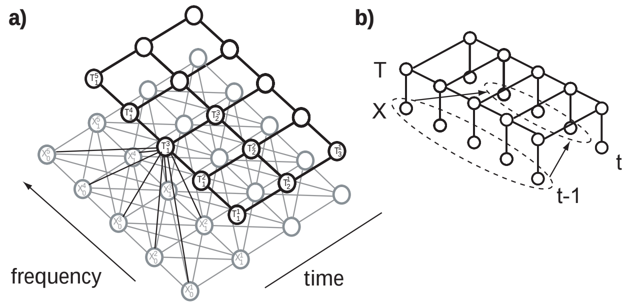

Fig. 4. a) Graphical model b) Graphical simplification.

Condition two.- The algorithm terminates once two messages have been passed over every edge, once in each direction. The marginal posterior probability for (hidden) variable node X_i, $p(X_i)$, is found by multiplying all the incoming messages into the node. Alternatively, it can be computed by multiplying the incoming and outgoing messages through the same edge. The operation of the algorithm is described for the case of an HMM on Appendix I.

E. Loopy Belief Propagation

The sum-product algorithm, like the junction tree algorithm, computes exact inference in models that can be organized as trees (fig. 2a), i.e. models without loops. When the model structure involves loops (fig.2b), *condition one* for the algorithm operation can not be met. In those situations, the sum-product algorithm can still be used although it no longer provides exact inference. There is evidence, however, that it approximates exact inference [10], [11]. When the sum-product algorithm is used for models with loops it is called the loopy belief propagation algorithm. The lack of clear leaf nodes in loopy graphs blurs the message passing initialization process, creating the need for *ad hoc* schedules for the message passing. Whenever *condition one* cannot be met, the missing incoming messages are set to be uniform, which requires iterating the message passing procedure even after *condition two* has been met. The algorithm should finish when the inference of the variables’ posteriors does not change between successive iterations (where an iteration is defined as a complete cycle of message passing rules). Given the existence of loops in the graph, there is no theoretical proof that such convergence can be reached; however, in practice, it does occur.

III. Spectral Deformation Model

Many audio signals, including speech and musical instruments, have short-time spectral representations that show great similarity between temporally-adjacent frames over much of the signal. We propose a model that discovers and tracks the nature of such correlation by finding how the patterns of energy are transformed between adjacent frames and how those transformations evolve over time.

Figure 3 shows a narrow-band spectrogram representation of a speech signal, where each column depicts the energy content across frequency in a short-time window, or time-frame. The value in each cell is actually the log-magnitude of the short-time Fourier transform in decibels:

$$x_t^k = 20\log\left(\text{abs}\left(\sum_{\tau=0}^{N_F-1} w[\tau]x[t\cdot H+\tau]e^{-j2\pi\tau k/N_F}\right)\right) \tag{12}$$

where t is the time-frame index, k indexes the frequency bands, N_F is the size of the discrete Fourier transform, H is the hop between successive time-frames, $w[\tau]$ is the N_F-point short-time window, and $x[\tau]$ is the original time-domain signal. We use 32 ms windows with 16 ms hops. Using subscript C to designate current and P to indicate previous, the model predicts a patch of N_C time-frequency bins centered at the k^{th} frequency bin of frame t as a “transformation” of a patch of N_P bins around the k^{th} bin of frame $t-1$, i.e.

$$\vec{X}_t^{[k-n_C,k+n_C]} \approx \vec{T}_t^k \cdot \vec{X}_{t-1}^{[k-n_P,k+n_P]} \tag{13}$$

where $n_C = (N_C-1)/2$, $n_P = (N_P-1)/2$, and $\vec{T}_t^k$ is the particular $N_C \times N_P$ transformation matrix employed at that point on the time-frequency plane. Figure 3 shows an example with $N_C = 3$ and $N_P = 5$ to illustrate the intuition behind this approach. The selected patch in frame t can be seen as a close replica of an upward shift of part of the patch highlighted in frame $t-1$. This “upward” relationship can be captured by a transformation matrix such as the one shown in the figure. The patch in frame $t-1$ is larger than the patch in frame t to permit both upward and downward motions. The proposed model selects, from a discrete set, the particular transformation that better describes the evolution of the energy from frame $t-1$ to frame t around every one of the time frequency bins x_t^k in the spectrogram. The patches used between adjacent time frequency bins overlap, which promotes transformation consistency [4]. The model also tracks the structure of the transformations throughout the whole signal to find useful patterns of transformation.

The generative graphical model is depicted in figure 4. Nodes $\mathcal{X} = \{x_1^1, x_1^2, ..., x_t^k, ..., x_T^K\}$ represent all the time-frequency bins in the spectrogram. For now, we consider the continuous nodes $\mathcal{X}$ as observed, although below we will allow some of them to be hidden when analyzing the missing-data scenario.

Discrete nodes $\mathcal{T} = \{T_1^1, T_1^2, ..., T_t^k, ..., T_T^K\}$ index the set of transformation matrices used to model the dynamics of the signal. Each $N_C \times N_P$ transformation matrix $\vec{T}$ is of the form:

$$\begin{pmatrix} \vec{w} & 0\,0 \\ 0 \;\; \vec{w} & 0 \\ 0\,0 & \vec{w} \end{pmatrix} \tag{14}$$

i.e. each of the N_C cells at time t predicted by this matrix is based on the same transformation of cells from $t-1$, translated to retain the same relative relationship. Here, $N_C = 3$ and $\vec{w}$ is a row vector with length $N_W = N_P - 2$; using $\vec{w} = (0\ 0\ 1)$ yields the transformation matrix shown in figure 3. To ensure symmetry along the frequency axis, we constrain N_C, N_P and N_W to be odd. The complete set of $\vec{w}$ vectors include upward/downward shifts by whole bins as well as fractional shifts. An example set, containing each $\vec{w}$ vector as a row, is:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & .25 & .75 \\ 0 & 0 & 0 & .75 & .25 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & .25 & .75 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ .75 & .25 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{15}$$

The length N_W of the transformation vectors defines the supporting coefficients from the previous frame $\vec{X}_{t-1}^{[k-n_W, k+n_W]}$ (where $n_W = (N_W - 1)/2$) that can "explain" x_t^k.

For harmonic speech signals sampled at 16 kHz and analyzed over 1024-point short-time windows (15 Hz bin resolution), we have found that a model using the above set of $\vec{w}$ vectors with parameters $N_W = 5$, $N_P = 9$ and $N_C = 5$ (which corresponds to a model with a transformation space of 13 different matrices $\vec{T}$) is very successful at capturing the self-similarity and dynamics of the harmonic structure.

The transformations set could, of course, be learned, but in view of the results we have obtained with this predefined set, we defer the learning of the set to future work. The results presented in this paper are obtained using the *fixed* set of transformations described by the matrix in eqn. 15.

Since we want to capture spectral transformations that can be described in the form of eqn. 13, we need to select potentials that impose such restrictions on the data. Therefore, the "local-likelihood" potential between the time-frequency bin x_t^k, its relevant neighbors in frame t, its relevant neighbors in frame $t-1$, and its transformation node T_t^k has the following form:

$$\psi\left(\vec{X}_t^{[k-n_C, k+n_C]}, \vec{X}_{t-1}^{[k-n_P, k+n_P]}, T_t^k\right) = \mathcal{N}\left(\vec{X}_t^{[k-n_C, k+n_C]}; \vec{T}_t^k \vec{X}_{t-1}^{[k-n_P, k+n_P]}, \Sigma^{[k-n_C, k+n_C]}\right) \tag{16}$$

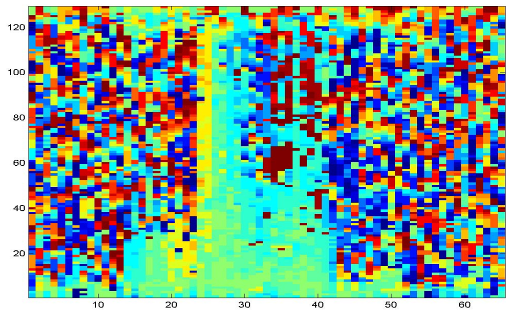

Fig. 5. Transformations that naively maximize the likelihood potentials. Each color represents a different transformation matrix from the set of 13.

The diagonal matrix $\Sigma^{[k-n_C,k+n_C]}$, which is learned, has different values for each frequency band to account for the variability of noise across frequency bands. Local constraints between adjacent transformation nodes are modeled by horizontal and vertical transition potentials $\psi_{hor}(T_t^k, T_{t+1}^k)$ and $\psi_{ver}(T_t^k, T_t^{k+1})$, which are represented by transition matrices.

A naive approach for finding the best set of transformations $\mathcal{T}$ that better describe the data would be to choose the transformations T_t^k that maximize the local potentials, eqn. 16. Figure 5a shows an example of such transformations for the spectrogram on the left in figure 6. In the figure each color indexes a different transformation matrix. There is little visible structure since the transformation choices capture only local information. If we require transformations with more global consistency we have to perform inference on the model.

When nodes $\mathcal{X}$ are fully observed, inference consists of finding probabilities for each transformation index at each time-frequency bin. Exact inference is intractable given that our model is quite "loopy", and it is approximated using Loopy Belief Propagation [10], [11]. Figure 7 shows the factor graph representation of a section of our model. For now, variable nodes x_t^k are observed, therefore all messages $m_{x_k^t \to g_i^j}$ consist of trivial identity messages; the only messages we need to compute are the ones that go through the T_t^k variable nodes.

Our schedule for the "belief propagation" is as follows: We first run messages through the vertical chains, i.e. all the bins in a given frame t. Next, we run messages through all the horizontal chains (constant frequency index). We choose this order because there is more correlation between the frequency bins in a given frame than between the bins at the same frequency across all time frames. Applying the "belief propagation" formulas on the chains results in forward/backward, upward/downward recursions similar to the ones obtained in HMMs. But unlike HMMs where the posterior at each point is determined by the local likelihood and by the neighbors in the chain, here the equations result in a weighted local likelihood that takes into account the match to the local observation as well as the "beliefs" from the neighboring chains. (This derivation is presented in Appendix II). The use of HMM-like recursions make the inference procedure relatively fast.

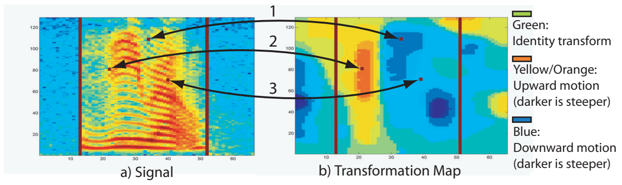

Fig. 6. Example transformation map showing corresponding points on the original signal.

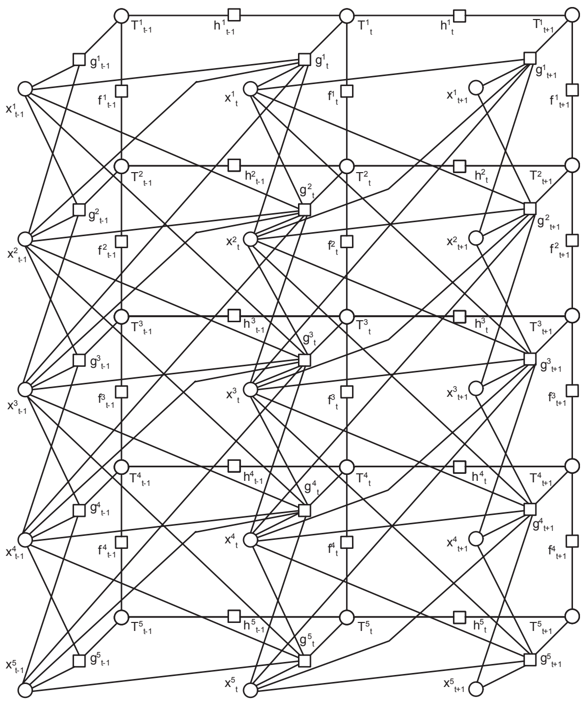

Fig. 7. Factor Graph for the relationships between spectrogram bins x_t^k and transformation nodes T_t^k.

We consider a full iteration of the model as a full pass of messages in both directions for all the vertical and horizontal chains. Given that the graph has loops we typically find it takes around five iterations before the transformations posteriors converge.

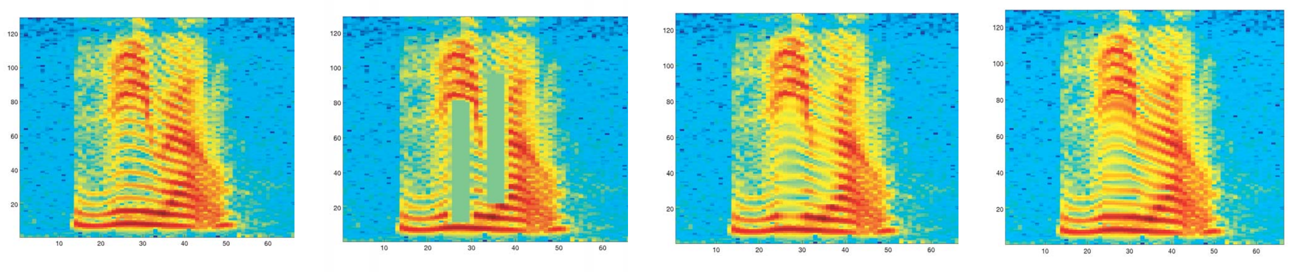

Fig. 8. Missing data interpolation example a) Original, b) Incomplete, c) After 10 iterations, d) After 30 iterations.

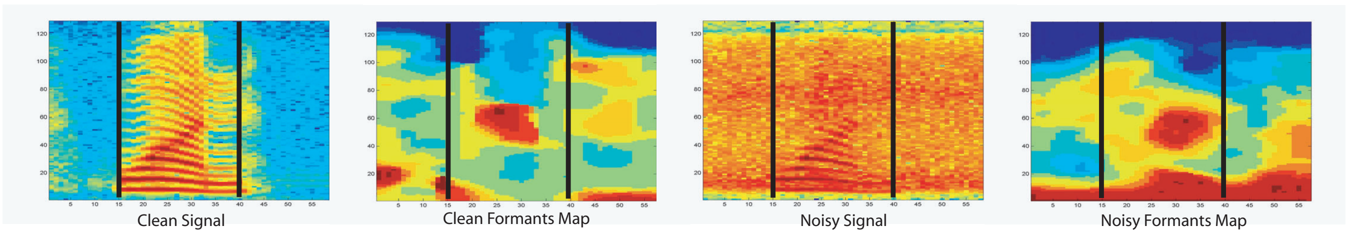

Fig. 9. Formant tracking map for clean speech (left panels) and speech in noise (right panels).

Once converged, we can find the transformation map, a graphical representation of the *expected* transformation node indices across time-frequency, which provides an appealing description of the harmonics' dynamics as shown in figure 6. In these panels, the links between three specific time-frequency bins and their corresponding transformations on the map are highlighted. Bin 1 is described by a steep downward transformation, while bin 3 also has a downward motion but is described by a less steep transformation, consistent with the dynamics visible in the spectrogram. Bin 2, on the other hand, is described by a steep upwards transformation. Notice how the transformation map emphasizes the global structure of the signal that the naive approach fails to reflect. Also, since the transformation maps follow global consistencies they can be robust to noise (see fig. 9), potentially making them a valuable representation in their own right, as investigated in section VIII for speech recognition.

IV. Inferring Missing Data

If a certain region of cells in the spectrogram is missing, as in the case of corrupted data, the corresponding nodes in the model become hidden. This is illustrated in figure 8, where regions of the spectrogram have been removed and tagged as missing. Inference of the missing values is performed again using belief propagation, although the update equations are more complex since there is the need

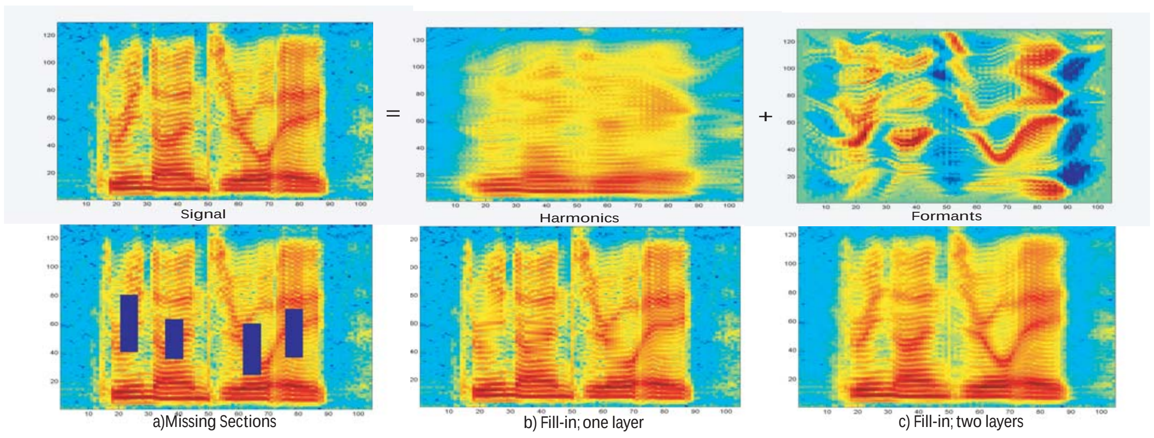

Fig. 10. First Row: Harmonics/Formants decomposition (posterior distribution means). Second Row: (a) Spectrogram with deleted (missing) regions. (b) Filling in using a single-layer transformation model. (c) Results from the two-layer model.

to deal with continuous messages.

The posteriors of the hidden continuous nodes are represented using Gaussian distributions, and the missing sections on figure 8b are filled in with the means of their inferred posteriors as shown in figure 8 parts c and d.

The transformation node posteriors for the missing region are also estimated. In the early stages of the "fill-in" procedure the transformation "belief" $(m_{g_j^i \to T_i^j})$ from the local likelihood potential g_i^j to the transformation nodes T_i^j that interact with 'missing" nodes $\tilde{x}_t^k$ are set to uniform so that their transformation posteriors are driven only by the reliable observed neighbors. Messages $m_{\tilde{x}_k^t \to g_i^j} = \delta(\tilde{x}_t^k - \mu)$ are initialized with the mean μ of the observed data.

The fill-in process starts with the missing values that have reliable immediate neighbors. Once those missing values have been filled-in with estimated data (i.e. using the mean μ_t^k of their Gaussian distributions) the process continues to their immediate "missed" neighbors and so on. Full details including the equations used in this scenario are given in Appendix III.

Here, an iteration is defined as each time the complete set of missing values is estimated. Define N_{width} as the maximum number of consecutive corrupted bins in any direction: The transformation posteriors are re-estimated every $N_{width}/3$ iterations, and at each re-estimation those transformation "beliefs" from the local likelihood potential g_i^j to the transformation nodes T_i^j that interact with 'missing" nodes x_t^k are recomputed using the newly-estimated values for the missing variables. This is important to ensure that the estimated data from different directions of the missing region agree. The algorithm normally converges after $N_{width} \times 3$ iterations.

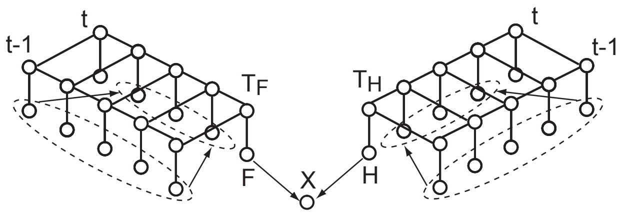

Fig. 11. Graphical representation of the two-layer source-filter transformation model.

V. Two Layer Source-Filter Transformations

Many sound sources, including voiced speech, can be successfully modeled as the convolution of a broadband *source excitation*, such as the pseudo-periodic glottal flow, and a time-varying resonant *filter*, such as the vocal tract, that 'colors' the excitation to produce speech sounds or other timbres. When the excitation has a spectrum consisting of well-defined harmonics, the resulting spectrum is in essence the resonant frequency response sampled at the frequencies of the harmonics, since convolution of the source with the filter in the time domain corresponds to multiplying their spectra in the Fourier domain, or adding in the log-spectral domain. Hence, we can model the log-spectra X as the sum of variables F and H, intended to be explicit models of the formants and harmonics of the speech signal. The source-filter transformation model is based on two additive layers of the deformation model described above, as illustrated in figure 11.

Variables F and H in the model are hidden, while X can be observed or hidden, as before. The symmetry between the two layers is broken by using different parameters in each, chosen to suit the particular dynamics of each component. We use transformations with a larger support in the formant layer ($N_W = 9$) compared to the harmonics layer ($N_W = 5$). Since all harmonics tend to move in the same direction, we enforce smoother transformation maps on the harmonics layer by using potential transition matrices with higher self-loop probabilities.

An example of the transformation map for the formant layer is shown in figure 9, which also illustrates how these maps can retain key features in the face of high levels of signal corruption; belief propagation searches for a consistent dynamic structure within the signal, and since noise is less likely to have a well-organized structure, it is properties of the speech component that are extracted. Inference in this model is more complex, but the actual form of the continuous messages is essentially the same as in the

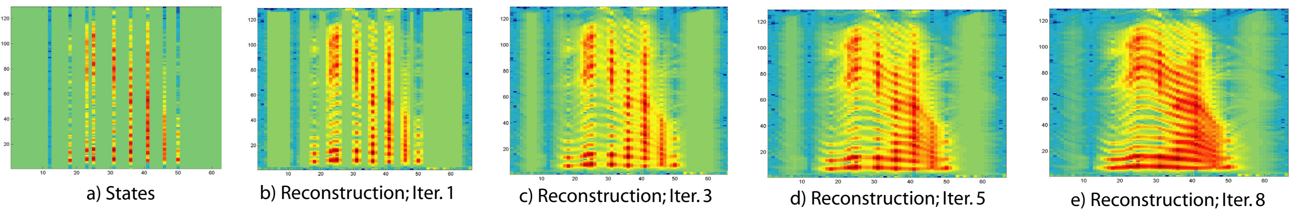

Fig. 12. Reconstruction from the matching-tracking representation, starting with just the explicitly-modeled states, then progressively filling in the transformed intermediate states.

one layer case, with the addition of the potential function relating the signal x_t^k with its transformation components at each time-frequency bin h_t^k (not to be confused with the factor graph nodes) and f_t^k:

$$\psi(x_t^k, h_t^k, f_t^k) = \mathcal{N}(x_t^k; h_t^k + f_t^k, \sigma^k) \tag{17}$$

The two layers are iteratively estimated as described on Appendix IV. An iteration is defined as one estimation of the harmonic layer followed by one estimation of the formants layer. The model usually converges after 10 iterations.

The first row of figure 10 shows the decomposition of a speech signal into harmonics and formants components, illustrated as the means of the posteriors of the continuous hidden variables in each layer. The decomposition is not perfect, since we separate the components in terms of differences in dynamics; this criteria becomes insufficient when both layers have similar motion. However, separation improves modeling precisely when each component has a different motion, and when the motions coincide, it is not really important in which layer the source is actually captured. In the second row of fig. 10, panel (a) shows spectrogram from the top row with deleted regions; notice that the two layers have distinctly different motions. In panel (b) the regions have been filled via inference in a single-layer model; notice that since the formant motion does not follow the harmonics the formants are not captured in the reconstruction. In panel (c) the two layers are first decomposed and then each layer is filled in; the figure shows the addition of the filled-in reconstructions from each layer.

VI. Matching-Tracking Model

Prediction of frames from their context is not always possible, for instance when there are transitions between silence and speech or transitions between voiced and unvoiced speech. As a result, we need a set of states to represent these unpredictable frames explicitly. We will also need a second "switch" variable that will decide when to "track" (transform) and when to "match" the observation with a state. Figure

13 shows a graphical representation of this model. At each time frame, discrete variables S_t and C_t are connected to all frequency bins in that frame. S_t is a uniformly-weighted Gaussian Mixture Model containing the means and the variances of each of the explicitly-modeled states, μ_j and ϕ_j. Variable C_t takes two values: when it is equal to 0, the model is in "tracking mode"; a value of 1 designates "matching mode".

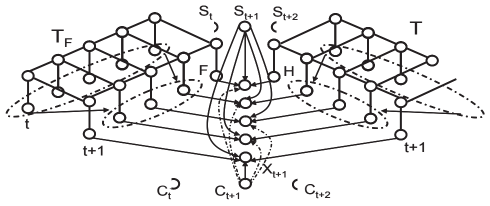

Fig. 13. Graphic model of the matching-tracking model

The potentials between observations x_t^k, harmonics and formants hidden nodes h_t^k and f_t^k respectively, and variables S_t and C_t, are given by:

$$\psi\left(x_t^k, h_t^k, f_t^k, S_t, C_t = 0\right) = \mathcal{N}\left(x_t^k; h_t^k + f_t^k, \sigma^k\right) \tag{18}$$

$$\psi\left(x_t^k, h_t^k, f_t^k, S_t = j, C_t = 1\right) = \mathcal{N}\left(x_t^k; \mu_j^k, \phi_j^k\right) \tag{19}$$

Inference is done again using loopy belief propagation. Defining ϕ as a diagonal matrix, the M-Step is given by:

$$\begin{aligned} \mu_j &= \frac{\sum_t Q(S_t = j)Q(C_t = 0)X_t}{\sum_t Q(S_t = j)Q(C_t = 0)} \\ \sigma_k &= \frac{\sum_t Q(C_t = 1)(x_t^k - (f_t^k + h_t^k))^2}{\sum_t Q(C_t = 1)} \\ \phi_j &= \frac{\sum_t Q(S_t = j)Q(C_t = 0)(X_t - \mu_j)^2}{\sum_t Q(S_t = j)Q(C_t = 0)} \end{aligned} \tag{20}$$

$Q(S_t)$ and $Q(C_t)$ are obtained using the belief propagation rules. $Q(C_t = 0)$ is large if eqn. 18 is larger than eqn. 19 for several time frequency bins at frame t. In early iterations when the means are still quite random, eqn. 18 is quite large, making $Q(C_t = 0)$ large with the result that the explicit states are never used. To prevent this we start the model with large variances ϕ and σ, which will result in non-zero values for $Q(C_t = 1)$, and hence the explicit states will tend to be learned.

As we progress, we start to learn the variances by annealing the thresholds i.e. reducing them at each iteration. We start with a relatively large number of means, but this becomes much smaller once the

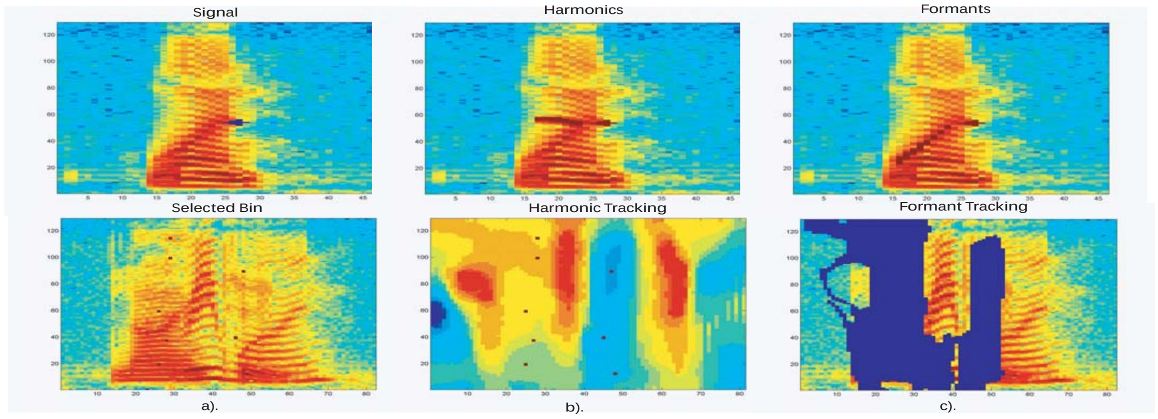

Fig. 14. Row 1: Harmonics/formants tracking example. The transformation maps on both layers are used to find the ancestors a given time-frequency bin (shown by the dark patches). Row 2: Semi-supervised two speaker separation. (a) The user selects bins on the spectrogram that she believes correspond to one speaker. (b) The system finds the corresponding bins on the transformation map. (c) The system selects and removes all bins whose transformations match the ones chosen; the remaining bins are assumed to correspond to the other speaker.

variances are reduced; the lower thresholds then control the number of states used in the model. The resulting states typically consist of single frames at discontinuities as intended. An iteration for this model consist of finding the posteriors for each one of the layers then applying the belief propagation rules to nodes S_t and C_t. Finally, the means and variances are learned through eqns. 18 and 19.

Figure 12a shows the frames chosen for a short speech segment whose spectrogram is shown in figure 8. The following panes show, at various iterations, how the signal can be regenerated from the model using the states and the two estimated motion fields. This reconstruction is another instance of inferring missing values, but in this case the motion fields are not re-estimated since we have the true ones.

VII. Model Demonstration

We have built an interactive demo that implements formant and harmonics tracking, missing data interpolation, formant/harmonics decomposition, and semi-supervised source separation of two speakers. Videos illustrating the use of this demo are available at: `http://www.ee.columbia.edu/~mjr59/def_spec.html`.

Formants and Harmonics Tracking: Analyzing a signal with the two-layer model permits separate tracking of the harmonic and formant ‘ancestors’ of any given point. The user clicks on the spectrogram to select a bin, and the system reveals the harmonics and formant “history” of that bin, as illustrated in the first row of figure 14.

Semi-Supervised Source Separation: After modeling the input signal, the user clicks on time-frequency bins that appear to belong to a certain speaker. The program then selects all neighboring bins with the same value in the transformation map; the remaining bins should belong to the other speaker. The second row of figure 14 depicts an example with the resultant mask and the "clicks" that generated it. Although far from perfect, the separation is good enough to perceive each speaker in relative isolation.

The demo also includes the missing data interpolation and harmonics/formants separation as described in the earlier sections.

Features	CLEAN	SNR20	SNR15	SNR10	SNR5
PLP12+delta	.94	2.3	4.1	7.9	12.2
PLP12+delta+dct8(FTM1)	.98	2.3	3.4	6.8	11.1
PLP12+delta+dct10(FTM1)	.99	2.3	3.3	7.4	11.3
PLP12+delta+dct8(FTM2)	1.3	2.5	4.2	9.7	12.5
PLP12+delta+dct10(FTM2)	1.3	2.5	4.2	9.4	12.7

TABLE I

WORD ERROR RATE PERCENTAGES OBTAINED WITH DIFFERENT SETS OF FEATURES AS A FUNCTION OF SIGNAL-TO-NOISE RATIO IN DB.

VIII. SPEECH RECOGNITION RESULTS:

The phonetic distinctions at the basis of speech recognition reflect vocal tract filtering of glottal excitation. In particular, the dynamics of formants (vocal tract resonances) are known to be powerful "information-bearing elements" in speech.

The formant transformation maps capture information about the global dynamics of the formants since the belief propagation algorithm searches for consistent structure in the energy evolution across both time and frequency. The delta and double-delta features of commonly-used features such as Perceptual Linear Prediction (PLP) or Mel Frequency Cepstrum Coefficients (MFCC) similarly capture local dynamics of energy, but they describe the frame-to-frame changes only within each frequency band. The transition maps can capture how the energy is moving *across* frequencies.

We computed two sets of transformation maps: one using formants obtained with our model as described above, and another with formants obtained using conventional cepstral smoothing. For the latter we only require a single layer model to compute the transformations maps. We then use features derived from these maps in combination with standard features in a speech recognizer to test if the maps can contribute new information not captured by the regular features. We chose PLP coefficients plus deltas as the baseline features.

To convert the formant transformation maps into features suitable for the recognizer, we applied mel-scale filtering to the maps then used a discrete cosine transform to decorrelate and further reduce the dimensionality of the final feature vectors.

We used the Aurora-2 noisy digits database for our experiments [**?**]. We trained on the complete "multicondition" training set and tested the recognizer using test set C (mismatched noises and channel). Results at different SNR levels are shown in table I. Features derived from formant transformation maps obtained using two layers are referred to as "FTM2" and the ones obtained from the single layer model are denoted "FTM1"; we tried using both 8 and 10 coefficients from the DCT ("dct8" and "dct10").

Looking at the results obtained using PLP features combined with FTM1 features (second and third rows in table I), we see that recognizer performance remains about the same as the standard features alone (first row) when the signal has high SNR values, but when the SNR decreases the new features improve the word error rate (WER) by as much as 19.5% relative for the 15 dB SNR ("SNR15") condition. We interpret these results as follows: when the signals are relatively clean, a local analysis of the energy dynamics, as performed by conventional features, is sufficient to effectively disambiguate the words. However as the interference becomes larger a more global model of the energy dynamics, such as the formants transition maps, can reduce the influence of local energy variations due to the noise. The belief propagation process searches for a consistent dynamic structure within the signal, and since noise is less likely to have a well-organized structure, it is the properties of the speech component that are extracted.

The table also shows that FTM2 features derived from the two-layer version of the model do not improve the performance of the recognizer. This may be because the layers cannot be separated when the two layers have parallel dynamics, as mentioned above: When the formants and the harmonics are well modeled by the same transformation, the formants are usually captured and modeled in the harmonics layer. Independent modeling of transformation maps for both layers may be more important for other applications such as the missing data inference in section IV, as well as the source separation approach described in the following section.

IX. Unsupervised Dominant Source Separation

The separation of speech mixtures into its individual sources using a single microphone is a very hard problem that has generated considerable interest in the research community. Current approaches include attempts to segregate a time-frequency representation (spectrogram) on a bin-by-bin basis, sometimes called time-frequency masking. Each bin is subjected to analysis and tagged as belonging to one of the individual sources. The large combinatorial space created by the analysis of the signal at such a fine resolution poses a great challenge to systems attempting to do such a separation. In [12] the combinatorial search is restricted by the use of pretrained speaker models, which limits the applicability of the approach to mixtures of sources whose individual properties in great detail. In [13], a training session is required to choose the right parameters for a spectral clustering algorithm. Finding clusters among the set of all time-frequency bins requires huge matrices that pose significant numerical problems. Hence, the algorithm requires a great deal of time to separate short mixtures. On the other hand, other research had shown that an intelligible separation can be done by grouping those regions of the spectrogram where a given speaker is more dominant than the others [**?**]. The problem is how to find those speaker-dominant regions.

A subband version of our matching-and-tracking model has been used to segment such regions, which can be further clustered to separate the sources. Since the number of these regions is significantly smaller than the total number of time-frequency bins in the spectrogram, the clustering problem is several orders of magnitude less complex. The process of segmenting the spectrogram of a mixture into its dominant speaker regions, examples of how to cluster those regions, and how to further infer masked (missing) regions will be described in a forthcoming paper.

X. Conclusions

We have presented a harmonic/formant separation and tracking model that models the short-time spectra of sound sources as local transformations of their immediate neighbors, then infers globally-consistent patterns of transformation through belief propagation. From among a range of applications, we demonstrate the usefulness of this information on a recognition task for speech in noise. The model has several other possible applications, including reconstructing partially-masked sounds and segmenting sounds into regions dominated by individual sources; we will examine these in more detail in the future.

APPENDIX I
SUM-PRODUCT ALGORITHM ON HMMS

Referring to figure 1 b), where $g_t = p(Y_t \mid X_t)$ and $h_t = p(X_{t+1} \mid X_t)$, then $m_{g_t \to X_t} = g_t$. The algorithm starts at the leaf nodes X_0 and X_T.

$m_{X_0 \to h_0} = m_{g_0 \to X_0} = g_0$, $m_{h_0 \to X_1} = \sum_{X_0} h_0 m_{X_0 \to h_0}$.

$m_{X_1 \to h_1} = m_{h_0 \to X_1} g_1 = \sum_{X_0} (h_0 m_{X_0 \to h_0}) g_1$.

Continuing forward the following recursion formula can be obtained:

$m_{X_t \to h_t} = \sum_{X_{t-1}} (h_{t-1} m_{X_{t-1} \to h_{t-1}}) g_t$

$m_{X_t \to h_t} = \sum_{X_{t-1}} (p(X_t \mid X_{t-1}) m_{X_{t-1} \to h_{t-1}}) p(Y_t \mid X_t)$.

which is the conventional forward recursion for HMMs, (α_t). From the other end:

$m_{X_T \to h_{T-1}} = g_T$, $m_{h_{T-1} \to X_{T-1}} = \sum_{X_T} (h_{T-1} m_{X_T \to h_{T-1}})$.

$m_{X_{T-1} \to h_{T-2}} = m_{h_{T-1} \to X_{T-1}} g_{T-1}$.

$m_{h_{T-2} \to X_{T-2}} = \sum_{X_{T-1}} (h_{T-2} m_{X_{T-1} \to h_{T-2}})$

$m_{h_{T-2} \to X_{T-2}} = \sum_{X_{T-1}} (h_{T-2} m_{h_{T-1} \to X_{T-1}} g_{T-1})$.

$m_{h_{T-2} \to X_{T-2}} = \sum_{X_{T-1}} p(X_{T-1} \mid X_{T-2}) p(Y_{T-1} \mid X_{T-1}) m_{h_{T-1} \to X_{T-1}}$.

The last recursion corresponds to the conventional backwards recursion, (β_t). Computing p(X_t) as the multiplication of the messages on the edge to h_t. p(X_t) = $m_{X_t \to h_t} m_{h_t \to X_t} = \alpha_t \beta_t$.

APPENDIX II
MESSAGES FOR THE SPECTRAL DEFORMATION MODEL WITH FULLY OBSERVED SPECTROGRAM

Referring to figure 7, variables x_t^k are observed so they only send identity messages, i.e. $m_{x_t^k \to g_l^i} = \delta(x_t^k - \hat{x}_t^k)$, where $\hat{x}_t^k$ is the actual observations at that time-frequency bin. Function nodes g_t^k represents the likelihood potential (Eq. 16), $h_t^k = \psi_{hor}(T_t^k, T_{t+1}^k)$ and $f_t^k = \psi_{ver}(T_t^k, T_t^{k-1})$. Working on the vertical chain at frame t, from variable T_t^1 to variable T_t^K for a spectrogram with K coefficients.

$m_{g_t^1 \to T_t^1} = g_t^1$, $m_{T_t^1 \to f_t^1} = g_t^1 m_{h_{t-1}^1 \to T_t^1} m_{h_t^1 \to T_t^1}$.

$m_{f_t^1 \to T_t^2} = \sum_{T_t^1} (f_t^1 m_{T_t^1 \to f_t^1})$.

$m_{T_t^2 \to f_t^2} = m_{f_t^1 \to T_t^2} g_t^2 m_{h_{t-1}^2 \to T_t^2} m_{h_t^2 \to T_t^2}$.

Making $g_t^{'k} = g_t^k m_{h_{t-1}^k \to T_t^k} m_{h_t^k \to T_t^k}$.

$m_{T_t^2 \to f_t^2} = \sum_{T_t^1} (f_t^1 m_{T_t^1 \to f_t^1}) g_t^{'2}$.

This corresponds to an upward (in frequency) recursion $\alpha_t^{'k}$ on the vertical chain at frame t using a “weighted” local likelihood function $g_t^{'k}$, which corresponds to the regular local likelihood function weighted by the “belief” of the adjacent vertical chains. A similar “weighted” downward recursion can be found examining the sequence of messages from variable T_t^T to variable T_t^1. Analogous "weighted" forward/backward recursions can be found while working with the horizontal chains.

APPENDIX III

CONTINUOUS MESSAGES FOR THE MISSING DATA SCENARIO

The local likelihood messages, i.e. $m_{g_t^k \to T_t^k}$, of the function nodes g_t^k that have any of the missing time frequency bins as arguments are initially set to uniform. For all others, $m_{g_t^k \to T_t^k} = g_t^k$. Once this initialization is done, the messages involving the transformation nodes are estimated as above, so that the "transformation beliefs" for the missing time frequency bins are driven only by the "beliefs" of the surrounding reliable neighbors and not by the unreliable local likelihood potentials.

Messages from the local likelihood potential functions g_t^k to missing time frequency bin x_j^i are functions in term of x_j^i. Therefore to facilitate the manipulation of the messages we need to express local likelihoods g_t^k as functions of an individual time frequency bin x_j^i.

The local likelihood potentials are defined as: $g_t^k = \mathcal{N}\left(\vec{X}_t^{[k-n_C,k+n_C]}; \vec{T}_t^k \vec{X}_{t-1}^{[k-n_P,k+n_P]}, \Sigma^{[k-n_C,k+n_C]}\right)$

That can be rewritten as:

$g_t^k = \frac{1}{\sqrt{2\pi\Sigma}} exp^{-\frac{1}{2}(Z_t^k)'\Sigma^{-1}(Z_t^k)}$, where

$Z_t^k = X_t^{[k-n_C,k+n_C]} - T_t^k X_{t-1}^{[k-n_P,k+n_P]}$, is a function of variables $(x_t^{k-n_C}, x_t^{k-n_C+1}, .., x_t^{k+n_C}, x_{t-1}^{k-n_P}, .., x_{t-1}^{k+n_P}, T_t^k)$;

Since column vectors $X_t^{[k-n_C,k+n_C]}$ and $X_{t-1}^{[k-n_P,k+n_P]}$ are concatenations of individual bins. We can express Z_t^k either as:

$Z_t^k = a^i x_t^i + X_t^{[k-nc,k+nc]} .* (1_{N1} - a^i) - T_t^k X_{t-1}^{[k-np,k+np]}$ or

$Z_t^k = -T_t^k b^i x_{t-1}^i - T_t^k (X_{t-1}^{[k-n_P,k+n_P]} .* (1_{N2} - b^i)) + X_t^{[k-n_C,k+n_C]}$ Where a^i and b^i are N_1 and N_2 column vectors with zeros in all positions excepting the one corresponding to x_t^i and x_{t-1}^i relative to vectors $X_t^{[k-n_C,k+n_C]}$ and $X_{t-1}^{[k-n_P,k+n_P]}$; 1_{N1} and 1_{N2} are N1 and N2 column vectors of ones and (.*) is the matlab pointwise vector multiplication.

Generalizing even further we can express Z_t^k as:

$Z_t^k = \alpha_j^i x_j^i - \beta_j^i(\mathcal{X}_{t,t-1}^{[k]}, T_t^k)$

Where $\mathcal{X}_{t,t-1}^{[k]} = [[X_t^{[k-n_C,k+n_C]}, X_{t-1}^{[k-n_P,k+n_P]}] \setminus x_j^i]$.

$$\alpha_j^i = \begin{cases} a^i & : \quad j = t \\ -T_t^k b^i & : \quad j = t-1 \end{cases}$$

and

$$\beta_j^i(\mathcal{X}_{t,t-1}^{[k]}, T_t^k = \begin{cases} X_t^{[k-nc,k+nc]} .* (1_{N1} - a^i) - T_t^k X_{t-1}^{[k-np,k+np]} & : \quad j = t \\ -T_t^k (X_{t-1}^{[k-n_P,k+n_P]} .* (1_{N2} - b^i)) + X_t^{[k-n_C,k+n_C]} & : \quad j = t-1 \end{cases}$$

The fill-in process starts with the missing values that have reliable neighbors. In general, a given missing bin x_j^i will exchange messages with Np function nodes g_{t-1}^l at frame $j+1$ and with Nc function nodes g_j^r at frame j. If g_t^k is one of such function nodes. Then the message from function node g_t^k to variable x_j^i has the form.

$$m_{g_t^k \rightarrow x_j^i} = [\sum_{T_t^k} \int_{\mathcal{X}_{t,t-1}^{[k]}} \frac{1}{C} exp^{\frac{1}{2}(\alpha_j^i x_j^i - \beta_j^i(\mathcal{X}_{t,t-1}^{[k]}, T_t^k))' \Sigma^{-1} (\alpha_j^i x_j^i - \beta_j^i(\mathcal{X}_{t,t-1}^{[k]}, T_t^k))} m_{T_t^k \rightarrow g_t^k} \prod_{y \in \mathcal{X}_{t,t-1}^{[k]}} m_{y \rightarrow g_t^k} dy] \quad (21)$$

Where j is either $t-1$ or t and $i \in [k - n_P, k + n_P]$ if $j = t-1$ or $i \in [k - n_C, k + n_C]$ if $j = t$. The individual time frequency bins y that belong to the set $\mathcal{X}_{t,t-1}^{[k]}$ are a collection of missing and observed variables. The ones that are observed have identity messages $m_{y \rightarrow g_t^k} = \delta(y - \hat{y})$, where $\hat{y}$ is the actual observed value, while the ones that are missing should have messages $m_{y \rightarrow g_t^k}$ equal to the multiplication of their own (N_C + N_P - 1) $m_{g_s^r \rightarrow y}$ messages (Eq. 21) coming from all the other function nodes g_s^r connected to variable y. Given the exponential complexity of such $m_{y \rightarrow g_t^k}$ messages, we approximate them by delta functions, i.e. $m_{y \rightarrow g_t^k} = \delta(y - \mu_y)$. Parameters μ_y are initially set to the mean of the observed data, these parameters are eventually estimated as explained below.

$m_{T_t^k \rightarrow g_t^k} = m_{h_{t-1}^k \rightarrow T_t^k} m_{h_{t+1}^k \rightarrow T_t^k} \; m_{f_t^{k-1} \rightarrow T_t^k} m_{f_t^{k+1} \rightarrow T_t^k}$ is simplified by making one the position of the "most-likely" transformation, $\hat{T}_t^k$, and zero all the others. Then Eq. 21 reduces to:

$m_{g_t^k \rightarrow x_j^i} = \frac{1}{C} exp^{\frac{1}{2}(\alpha_j^i x_j^i - \hat{\beta}_j^i)' \Sigma^{-1} (\alpha_j^i x_j^i - \hat{\beta}_j^i)}$,

where $\hat{\beta}_j^i = \beta_j^i(\hat{\mathcal{X}}_{t,t-1}^{[k]}, \hat{T}_t^k)$. $\hat{\mathcal{X}}_{t,t-1}^{[k]}$ is formed by the concatenation of the relevant $\hat{y}$s and μ_ys.

The posterior probability of node x_i^j, $p(x_t^k)$, is equal to the multiplication of all its incoming messages. We approximate this multiplication with a Gaussian distribution, $q'(x_i^j) = \mathcal{N}(x_i^j; \mu_{x_i^j}, \phi_{x_i^j})$. Minimizing their KL divergence we find:

$$\mu_{x_i^j} = \frac{\sum_{l=1}^{N_C+N_P} \alpha_l' \Sigma_l^{-1} \hat{\beta}_l}{\sum_{i=1}^{N_C+N_P} \alpha_l' \Sigma_l^{-1} \alpha_l^{-1}} \quad (22)$$

The values displayed by the missing data application are these mean values. The mean of the variable to local function nodes messages, ($m_{y \rightarrow g_t^k} = \delta(y - \mu_y)$ for missing variables y in Eq. 21), have the same form as in equation 22, just subtracting the numerator and denominator factor corresponding to the incoming message from the corresponding function.

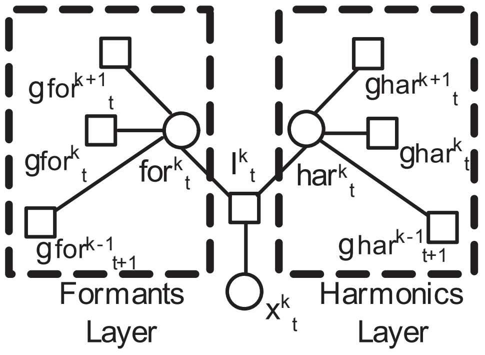

Fig. 15. Factor graph of a section of the two-layers model

Appendix IV
Two layers decomposition

We first estimate the harmonics layer. Initializing message $m_{l_k^t \to har_t^k}$ as $\mathcal{N}(l_k^t, x_t^k, \sigma^k)$, (figure 15 where l_k^t is the function node for the two layers local likelihood potential (17) and messages $m_{har_t^k \to ghar_k^t}$ = $\delta(l_k^t - x_t^k)$ with the actual values of the spectrogram. Then we estimate the posterior probabilities of the harmonics layer q(har_k^t) and their means μhar_t^k as in the one layer case using Eq. 22, adding to both, the denominator and numerator the corresponding terms from $m_{l_k^t \to har_t^k}$. Messages $m_{har_t^k \to ghar_k^t}$ are also recomputed. We then proceed to estimate the formants layer, initializing message $m_{l_k^t \to for_t^k}$ as $\mathcal{N}(l_k^t, x_t^k - \mu har_t^k, \sigma k)$, and messages $m_{for_k^t \to gfor_t^k}$ = $\delta(l_k^t - (x_t^k - \mu har_t^k)$ with the subtraction of the estimated harmonics layer from the observed data. The idea here, is to model in this layer all the data that was not captured by the harmonic layer given the restrictions on its parameters. We then estimate the posterior probabilities of the formants layer q(for_k^t) and their means μfor_t^k as in the case of the harmonics layer. Messages $m_{for_k^t \to gfor_t^k}$ are also recomputed. We then go back to re-estimate the harmonics layer, since all messages have been computed at least once, we just update the messages and recompute the harmonics layer means. Then the formants layer is re-estimated. We keep iterating back and forth until convergence.

Acknowledgment

This work was supported by Microsoft Research and by the NSF under grant no. IIS-0238301. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF

References

[1] S. Roweis, “One-microphone source separation,” in *Advances in NIPS*. Cambridge MA: MIT Press, 2000, pp. 609–616.

[2] J. Bilmes, “Data-driven extensions to hmm statistical dependencies,” in *ICSLP*, 1998.

[3] N. Jojic and B. Frey, “Learning flexible sprites in video layers,” in *CVPR*, 2001.

[4] B. F. N. Jojic and A. Kannan, “Epitomic analysis of appearance and shape,” in *ICCV*, 2003.

[5] A. Z. A. Levin and Y. Weiss, “Learning to perceive transparency from the statistics of natural scenes,” in *NIPS*, 2002, 2003.

[6] N. J. M. Reyes-Gomez and D. Ellis, “Towards single-channel unsupervised source separation of speech mixtures:the layered harmonics/formants separation-tracking model,” in *SAPA*, Korea, 2004.

[7] ——, “Deformable spectrograms,” in *AISTATS*, Barbados, 2005.

[8] M. Jordan, “Learning in graphical models.” Cambridge MA: MIT Press, 1999.

[9] B. F. F. Kschischang and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on information theory*, vol. 47, 2001.

[10] W. F. J.S. Yedidia and Y. Weiss, “Understanding belief propagation and its generalizations,” in *Exploring Artificial Intelligence in the New Millennium*, 2001.

[11] Y. Weiss and W. Freeman, “Correctness of belief propagation in gaussian graphical models of arbitrary topology,” *Neural Computation*, vol. 13, pp. 2173–2200, 2001.

[12] S. Roweis, “Factorial models and refiltering for speech separation and denoising,” in *Proc. EuroSpeech*, Geneva, 2003.

[13] F. Bach and M. Jordan, “Blind one-microphone speech separation: A spectral learning approach,” in *NIPS*, Vancouver, 2004.