2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Hierarchical Linear Dynamical Systems: A new model for clustering of time series

Goktug T. Cinar, Carlos A. Loza and Jose C. Principe

***Abstract*— The auditory cortex in the brain does effortlessly a better job of extracting information from the acoustic world than our current generation of signal processing algorithms. The proposed architecture, Hierarchical Linear Dynamical System (HLDS), is based on Kalman filters with hierarchically coupled state models that stabilize the input dynamics and provide a representation space. This approach extracts information from the input and self-organizes it in the higher layers leading to an algorithm capable of clustering time series in an unsupervised manner. In this paper we further investigate the properties of HLDS, demonstrate its performance on music rather than isolated notes and propose the time domain implementation to overcome one of its current bottlenecks.**

***Index Terms*— Music information retrieval, kalman filters, dynamical systems, hierarchical systems, cognitive models, clustering, time series**

I. Introduction

Clustering of time series is a challenging problem compared to its counterpart on static data sets. Therefore a lot of research effort have been put into this area. In [1] Liao surveys the current state of the art and reveals that most of the current work either modifies the distance measures used in clustering of static data to work on time series or extracts features from time series and applies methods of static data clustering on them. Our approach falls under the subgroup that uses modelling techniques to cluster time series as will be explained below, and is substantially different from the majority of the methods dealing with clustering of time series. Mainly because it takes advantage of the temporal information buried in the temporal structure of time series.

The signal processing and machine learning communities do not favor state space models and dynamical systems. This could be a poor decision when working on sensory stimulus as the stimuli is rich in terms of context at a given time (spatial information); however considerably large amount of information lies in the temporal behavior of the sensory input [2]. This is even more important when working with auditory stimuli, as we are working with a one-dimensional signal, most of the information can be gathered by analyzing the temporal structure. Therefore the bag of audio features representation, which is a common approach in music information retrieval, is fundamentally limited by ignoring the time dependency between feature vectors [3].

Goktug T. Cinar, Carlos Loza and Jose C. Principe are with the Computational NeuroEngineering Laboratory (CNEL), University of Florida, Gainesville, FL (email: gcinar@ufl.edu, cloza@ufl.edu, principe@cnel.ufl.edu).

This work is supported by the Office of Naval Research (ONR) grant #N000141010375.

The important question of finding good features for sensory processing remains unanswered. Hence we almost exclusively keep using the sensory space to find them. This may not be the best approach due to the complexity and variability of the sensed signals (changes in timbre, tone, noise, context), while what is needed is invariant representations. On the other hand, biological organisms have solved this problem long ago by developing an active perception mechanism. The brain disambiguates the sensory signals according to our expectations and prior knowledge.

Using the theory of dynamical systems to process and analyze sensory stimulus is increasingly becoming an area of interest [4], [5], [6]. Recently Barrington et al. [3], [7] proposed the use of Dynamic Texture (DT) to model music. Using Linear Dynamical Systems (LDS) as a generative model for time-series is also known as Dynamic Texture in the computer vision literature [3]. Barrington et al. used the Mixture of Dynamic Textures introduced in [8] to segment a song into sections. They use larger time scales to learn textures corresponding to particular segments in a song. Our focus will be to start from smaller time scales and expand the representations in time (thus changing slowly compared to the lower layer representations) while we build a hierarchical network, which is a property called *temporal coherence* [9]. Similarly, Vaizman et al. [10] used Mixtures of Dynamic Textures to extract features describing the emotional content in music. Also in [11] Coviello et al. used Mixtures of Dynamic Textures to perform automatic music tagging and music annotation.

Hopfield used recurrent artificial neural networks as a content addressable memory and demonstrated some properties including generalization and familiarity recognition [12]. Later Tank and Hopfield used these networks on speech signals. As they put it, “recognizing the pattern in a time-dependent signal is important in hearing and vision”. They introduce the idea of attractors in the ‘energy space’ for which “the presentation of a known sequence of stimuli builds a deep pit on the space-time energy surface with a wide valley leading to it” [13], [14]. In [13] they state that they would need a $n+1$ dimensional energy surface consisting one dimension for each of n recognition variables and one time dimension. This would essentially make the model computationally too expensive as the number of recognition variables grow. We share the same vision of addressing the possible need for a hierarchical set of networks for detecting sequences on several time scales. They demonstrate the model on recognition of digits in connected speech [15], where they propose a framework that consists

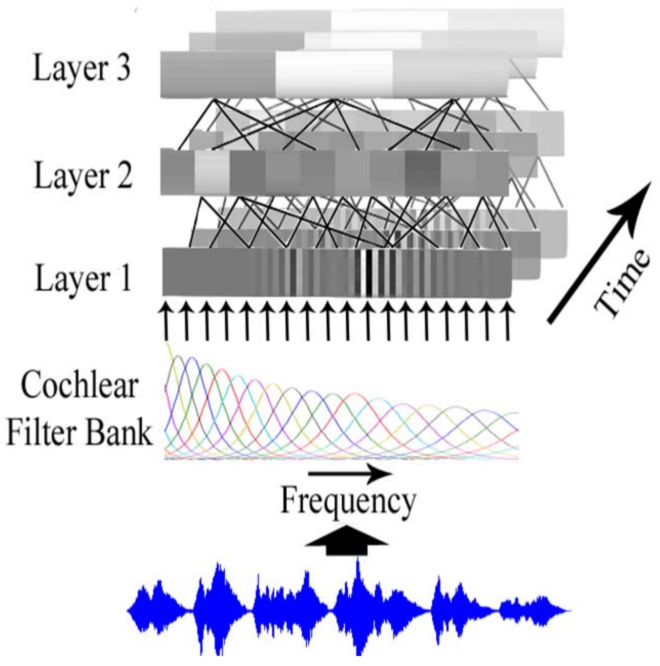

Fig. 1. The schematic showing the structure of the model.

of tone detectors and networks that work using the output of tone detectors to further recognize sequences.

In [16] Munkong and Juang present an overview of auditory perception process and advocate for the integration of structural and functional knowledge of the auditory system. They "suggest a perspective, called the constructionist's paradigm, which looks at the neuron-anatomical structure for auditory perception in three essential stages with intermediate signal representations that allow interchangeable coupling to accommodate new and improved models within each individual stage". We follow this paradigm in creating a model that works directly with auditory stimuli in its lowest layer and slowly builds a hierarchical representation with intermediate stages where key tasks can be performed.

The biological inspirations to build our model are limited and are restricted to guidelines. The model should have a hierarchical structure to represent the layered structure of the auditory cortex. It should have the ability to be (directly or indirectly) driven top-down by causes to mimic active perception. Therefore we will concentrate on Hierarchical Linear Dynamical Systems (HLDS). The overview of the model is shown in Figure 1.

In [17], we introduced the HLDS and presented the initial findings. In this paper we present additional experimental results as well as the extension of the model to work with time domain observations instead of the frequency domain used in [17].

II. Hierarchical Linear Dynamical System

It is a common belief that the cortex is stereotyped in hierarchical layers and internal organization [18]. This is the main motivation for choosing a Hierarchical Linear Dynamical System.

The idea of a nested HLDS is to design a model that would consist of one measurement equation and multiple state transition equations. The system is nested in the sense that each state transition equation creates the causes/states that would drive the lower layer. This introduces a top-down flow of information. In the nested HLDS we drive the system bottom-up by the observations, and top-down by the states. The governing equations of the nested HLDS are as follows:

$$\begin{aligned} z_t &= z_{t-1} + p_t \\ u_t &= G u_{t-1} + D z_{t-1} + r_t \\ x_t &= F x_{t-1} + B u_{t-1} + w_t \\ y_t &= H x_t + v_t \end{aligned} \qquad (1)$$

where $\mathbf{y}_t \in \mathbb{R}^m$ is the observation vector. $\mathbf{x}_t \in \mathbb{R}^n$ is the first layer hidden states, $\mathbf{u}_t \in \mathbb{R}^k$ is the second layer hidden states, and $\mathbf{z}_t \in \mathbb{R}^s$ is the third layer hidden states where $n > k > s$ so that the dimensionality decreases in each layer as we go up in the hierarchy. The motivation is to restrict the states in to smaller representation spaces to be used in clustering. In other words, we want the model's representation to expand as we go down the hierarchy. The model is shown to work with a wide range of model sizes in Section III. In these equations, p_t, r_t, w_t, v_t are zero mean Gaussian uncertainties. The covariance matrices of these uncertainties are defined as $\alpha_m I_m, \alpha_n I_n, \alpha_k I_k, \alpha_s I_s$ respectively where I_m is an identity matrix of dimension $m \times m$.

The reader should note that we impose a fixed point behavior (i.e. identity state transition) in the highest layer of the hierarchy. This combined with the locally stationary behavior of music signals imposes a stable behavior throughout the hierarchy as each layer is driven by the one above it. It is intuitive to think that even if slightly larger changes occur in the observations, the changes in the higher layers should be much slower. This should result in creating clusters in the state space that $\mathbf{z}_t \in \mathbb{R}^s$ exists.

A. State Estimation in Joint Space

We can re-write the nested dynamics defined in (1) as follows:

$$\tilde{X}_t = \tilde{F}\tilde{X}_{t-1} + \tilde{W}_t \qquad (2)$$

$$y_t = \tilde{H}\tilde{X}_t + v_t \qquad (3)$$

$$where \quad \tilde{X}_t = \begin{bmatrix} z_t \\ u_t \\ x_t \end{bmatrix}, \tilde{F} = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ D & G & \mathbf{0} \\ \mathbf{0} & B & F \end{bmatrix},$$

$$\tilde{H} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & H \end{bmatrix}, \tilde{W}_t = \begin{bmatrix} p_t \\ r_t \\ w_t \end{bmatrix}$$

The equations (2) and (3) tells us that there is a joint state-space where we can do the estimation of all the hidden states in all the layers simultaneously. This will enable the use of the standard estimation equations of the Kalman Filter [19] for this joint state space model.

B. Parameter Learning

We learn the parameters simultaneously while inferring the states of the HLDS. This is known as sequential estimation ([20], [21]), where we consider two dual systems with the same observation. The idea is as simple as switching the roles of the parameters and states, where we vectorize the parameters in $\left[\tilde{\mathbf{F}}, \mathbf{H}\right]$ and treat them as states. We consider identity state transition for parameter dynamics. We use gammatone filters in the first layer. We create 4^{th} order gammatone filters in the range from 10Hz to $f_s/2$ (where f_s is the sampling frequency of the recordings we use). We place the center frequencies half an ERB away from each other, to get a denser coverage in the frequency domain. Therefore in this paper the models all have a 60-dimensional first layer (n=60).

C. Parallelization of Parameter Learning

In [17] we have shown that the parameter learning can be parallelized with no loss and without any approximation. This saves a lot of computation and memory as the dual system for the parameters require working with very high dimensional systems. Here we present the parallelization briefly:

$$F_{n\times n} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \cdots \\ \mathbf{f}_n \end{bmatrix} \quad B_{n\times k} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \cdots \\ \mathbf{b}_n \end{bmatrix}$$

$$(\mathbf{x}_t)_{n\times 1} = \begin{bmatrix} x_t^1 \\ x_t^2 \\ \cdots \\ x_t^n \end{bmatrix} (\mathbf{u}_t)_{k\times 1} = \begin{bmatrix} u_t^1 \\ u_t^2 \\ \cdots \\ u_t^k \end{bmatrix}$$

where $\mathbf{f}_i, \mathbf{h}_i$,$\mathbf{b}_i$ is the i^{th} row of the corresponding matrix, x_t^i is the i^{th} element in the state vector.

To learn the values in $\mathbf{F}$ and $\mathbf{B}$ consider the following system:

$$\begin{aligned} \theta_t &= \theta_{t-1} + w_t^\theta \\ x_t^i &= \begin{bmatrix} x_{t-1} \\ u_{t-1} \end{bmatrix}^T \theta_t + v_t^\theta \quad \textit{where} \quad \theta_t = \begin{bmatrix} \mathbf{f}_i^T \\ \mathbf{b}_i^T \end{bmatrix} \end{aligned}$$

The high dimensionality of the parameter space was keeping us from propagating the error covariance of parameter estimation. Now with these parallel but much smaller systems the propagation can be done easily. The same initialization explained above is used for the dual systems estimating the parameters.

A common problem in applying the sequential estimation is the degenerate solution as $t \to \infty$,$\hat{\mathbf{w}}_t \to \infty$ and $\mathbf{x}_t \to 0$. To avoid this solution we force each column of the $\mathbf{w} = [\mathbf{F}, \mathbf{H}, \mathbf{D}, \mathbf{B}, \mathbf{G}]$ matrices to have unit norm. Therefore after each update, the values of each column are normalized such that the column has unit norm. Equation (4) shows the constraint we enforce.

$$\|F_t(.,j)\| = 1 \quad \forall j \tag{4}$$

III. Experimental Results

In [17], we worked in the frequency domain. Here we will recap the experimental setup we have used. Our observation sequences will be single-sided magnitude spectrum of the audio signals; we will not work with the phase of the spectrum. The audio signal will be windowed using a L-point symmetric Hanning window. The length of the window will be selected such that each window will contain about 80-100ms of data. The windows will have 50% overlap. Each window of data will be normalized to a fixed maximum amplitude before the FFT is computed.

In this experiment we use the audio samples from *The University of Iowa Musical Instrument Samples* [22]. All the instrument recordings that are used in this work are recorded in an anechoic chamber. A Neumann KM 84 Cardioid microphone is used at a distance of 5 feet. The mono recordings are sustained notes of around 2 seconds sampled at 44.1 kHz with 16-bit resolution. In our work the recordings have been downsampled to 11025 HZ mono so that the window size for the FFT would be reasonable. We used concatenated notes in the range E3-D6 for the non-vibrato B-sharp Trumpet to train and test the HLDS.

We use a 1024 point FFT to create the observation vectors from each window of data. As we use the single sided amplitude spectrum, the length of our observation vector is 512 (m=512). We present each observation vector multiple times to speed up convergence to the clusters. We have n=60 states in the first layer (the number of filters needed to cover the frequency range placed half an ERB apart), k=10 (chosen heuristically) states in the second layer and s=3 (for visaualization purposes) state in the third layer. We use $0.01 \cdot I_{(n+k+s)\times(n+k+s)}$ as the covariance matrix of the joint state transition equation (2) uncertainty; and $0.5 \cdot I_{m\times m}$ as the covariance matrix of the joint measurement equation (3) uncertainty.

Once the training is completed the parameters are fixed and the system is tested with trumpet notes. Figure 2 shows the point clusters learned by the HLDS. As explained in [17], we see that there is a transition phase before the states converge around the cluster. The system usually reaches the neighborhood of the cluster within a few (3-6) iterations. We observe that whatever the initial states are, they converge to the particular clusters when a certain note is played.

A. Monophonic Pitch Estimation

For pitch estimation using the clusters in the highest layer, we need to label cluster centers for each note. Therefore to determine the cluster centers we present the notes one-by-one to the system with its parameters fixed. As the transition phase between the notes is considerably short, we do not bother

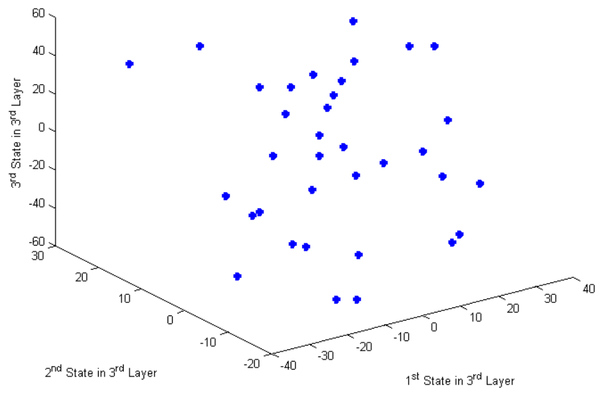

Fig. 2. The response of the 3^{rd} layer hidden states to a subset of the trumpet notes. The transients are not shown to clearly show the clusters.

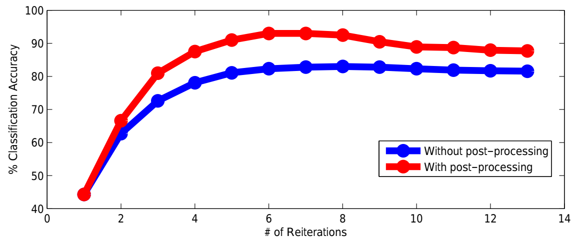

Fig. 3. The figure shows the curves of the pitch estimation accuracy with and without post-processing for different number of iterations over each window of music. Dimensions of the model are s=3, k=10, n=60.

eliminating the transition phase while determining the cluster centers. We find the mean value for the highest layer states for all time instances, and assign this value as the cluster center.

To assess the classification accuracy we do Monte Carlo runs through all 35 notes. In each run the notes are presented in a different order. At each time instant the states are compared to the cluster centers for each note. The closest center is assigned as the label. When a new note is presented we create a window or 'memory' of instantaneous labels. We then check if the same decision was given by the system for all time instances in the memory. When the unanimous decision is given for the window, convergence is declared. The memory size should be decided with care as too long of a memory would prevent us from recognizing notes that are played for a short duration. On the other hand too short of a memory can signal wrong convergence. We use a memory of 4 instantaneous labels. Using this method we have shown that the model consistently achieves over 90% classification accuracy, where as high as 96.6% accuracy was achieved for all time instances. The model was shown to have performed similarly for a large selection of dimensionalities for second and third layers as well. Results have been detailed in [17].

B. Performance in Music

We have tackled the problem of classifying the isolated notes successfully. However this setup is not 'practical'. In a real life scenario it would be almost impossible to find a piece that consists of notes sustained for at least 2 seconds. However we also didn't want to tackle the problem of dealing with different playing styles such as vibrato. Therefore we have created the 'Yesterday' from the Beatles, [23], using the notes that are in the database. This created a more challenging scenario to test the system compared to the isolated note classification. The main reason being the existence of notes with much shorter durations.

We present the music generated using the samples from our database and let the model run through the whole length with fixed parameters. At each time instant we assign the class to the current window w.r.t. the value of the highest layer states. We compare the current position of the state to the cluster centers and assign the closest center as the current class as we did in the supervised pitch estimation problem. As we created the music recording using the samples in the database, we have the ground truth for the class labels (note assignments) of each window. We evaluate the success of the algorithm w.r.t. the pitch estimation accuracy over all time instances.

We notice that most of the errors are concentrated around the onset of each note. This is caused by the wrong assignments the model makes before converging to the true cluster. To overcome this issue we decided to keep a window of note assignments in memory and look for consistent note assignments before making any decisions. We constantly monitor this sliding window of assignments and start labeling 'undecided' when one or more of the note assignments are different. After this event we wait for the window to have a unanimous decision. Once the unanimous decision is given, we go back in time and correct the assignments with the consistent assignment.

We have noticed that the convergence time of the system is the main bottleneck that we are facing. Figure 3 shows the curves of the classification accuracy with and without post-processing for different number of iterations over each window of music. After we perform the post-processing the classification accuracy goes as high as 93%. We couldn't break this barrier because the misclassified notes were shorter than the convergence time. This prompted us to investigate the convergence time of the system.

C. Convergence Time of the Model

We investigate the convergence time of the model experimentally. We randomly pick two notes and then play the first one. When we present the second note, we measure the time it takes for the model to start giving consistent decisions. This is repeated 500 times for two different model sizes to create a distribution of convergence times. We picked two well-trained models to use in this experiment. They both had over 90% classification accuracy for all time instances. So we expected a peak for the distribution for smaller number of windows. We see in Figure 4 that the distributions peak around 2-5

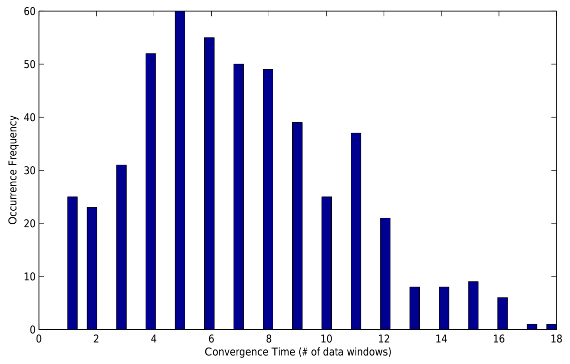

Fig. 4. Histogram of Convergence Times. Dimensions of the model are s=3, k=10, n=60.

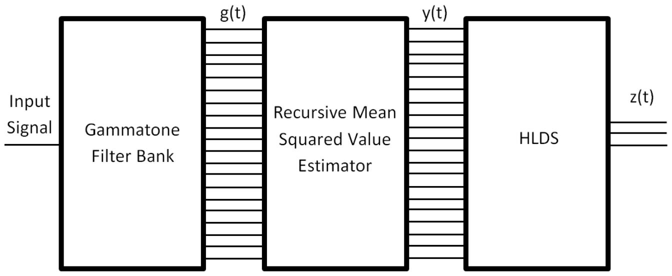

Fig. 5. The flow diagram of the time domain implementation.

windows (≈110-220 ms) as expected. As we wait for a few more windows of consistent decisions to make sure of the convergence after the first appearance of the cluster label, it takes the model around 6-10 windows to make a decision. Therefore the model needs about 250-400 ms to make a decision, which may be reasonable as "latencies as long as 0.5 seconds following brief auditory stimulation have been measured for neurons in A1 cortex of behaving monkeys" [13]. In the worst case the model takes about 16 windows of data to converge from one note to the other, which corresponds to about a second. These are the bottlenecks of the model regarding the pitch estimation task. This also explains why we couldn't go over certain accuracy for music transcription. Therefore we propose a time domain method to overcome the "downsampling" introduced by the spectral windowing. As we have considerably larger amount of data when working in time domain, we expect to reduce the effect of convergence time on the classification accuracy.

IV. Time Domain Approach

As the model requires a locally stationary observation sequence we will apply the following preprocessing steps to the time domain signals. We implement in the time domain the gammatone filters detailed above and pass the signal through 60 filters. The outputs are then passed through a recursive filter to estimate the mean squared values of the filter outputs:

$$y_t^i = \alpha(g_t^i)^2 + (1-\alpha)y_{t-1}^i \qquad (5)$$

TABLE I
The table shows the pitch estimation accuracy for all time instances of the algorithm implemented in time domain.

s=3, k=10, α=0.005	s=4, k=12, α=0.005	s=5, k=15, α=0.005	s=3, k=10, variable α
96.37%	91.8%	97.46%	98.73%

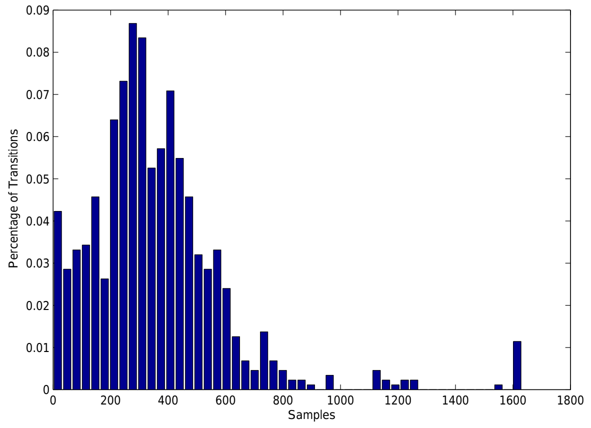

Fig. 6. Histogram of Convergence Times. Dimensions of the model are s=3, k=10, n=60.

where y_t^i is the i^{th} entry of the observation vector and g_t^i is the output of i^{th} filter at time t. Then the output of the recursive mean squared estimator is fed into the HLDS to train the model. The flow diagram is shown in Figure 5.

Following the same procedures we have trained and tested the models on three different model sizes. The results are summarized in Table I. We see that the performance is significantly better compared to our frequency domain implementation. Only one of the models performed closer to the frequency domain implementation as some of the clusters were too close in the 3^{rd} layer which caused the drop in accuracy. The convergence time of the model is analyzed similarly and a histogram of convergence times are shown in Figure 6. We see that it takes less than ≈ 50ms for the algorithm to converge in general which is significantly shorter compared to our frequency domain implementation.

When a large α parameter is selected the mean squared estimate will reach the neighborhood of the true value quickly but will have oscilaltions around it. On the other hand a small α will increase the convergence time of the estimate while minimizing the oscillations. A smaller α will in return increase the convergence time to the clusters in HLDS but will not suffer from the oscillations (which manifests themselves as orbits around the cluster centers). This is demonstrated in Figure 7. We see that the orbit size increases whereas the convergence time decreases for larger α.

In order to provide a more consistent model, we decided to optimize the free parameter of the mean square estimator, i.e. α, by selecting the particular value that would yield to

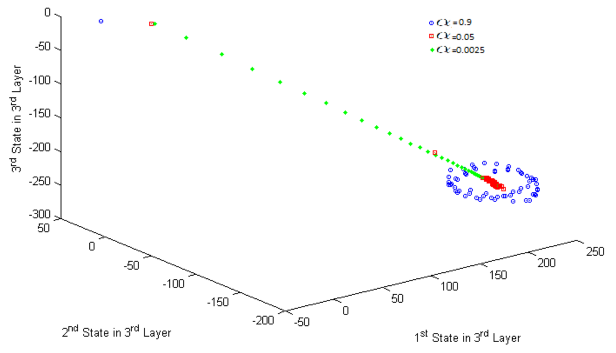

Fig. 7. The figure shows the effect of the α parameter on the model behavior.

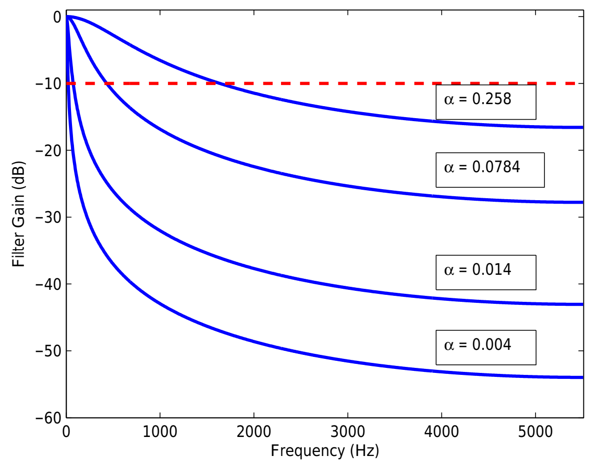

Fig. 8. The figure shows the frequency gain of the filters adjusted by alpha for a selection of the filters.

a 10% decay in the magnitude of the signal after having passed through the IIR filter implemented by the mean square estimator 5. In particular, we selected the central frequency of each one of the 60 gammatone filters and searched for the value of alpha that would provide a 10% decay (-10 dB) in the magnitude of the frequency response of the IIR filter at that particular central frequency. Figure 8 provides a graphical interpretation of the procedure for 4 central frequencies of the gammatone filters along with their corresponding α's. By doing this, we assured that the size of the clusters are consistent and relatively equal in diameter. This property could be particularly useful when dealing with more features (notes) present in the time series. Using these optimized α values we trained a model (s=3, k=10, n=60) and achieved 98.73% accuracy which outperformed the results obtained by fixed α.

V. Discussion and Conclusion

This paper shows that we have successfully formed a hierarchical model that extracts information and self-organizes this extracted information in its higher layers. This self organization leads to an algorithm capable of clustering time series in an unsupervised manner. This is substantially different from the majority of the methods dealing with clustering of time series as they do not take advantage of the temporal information. Here we simply take advantage of the internal dynamics of the presented signals and exploit them to embed the time series in smaller subspaces. When we look at the equations, we can notice that there is an observable subspace for each layer of states given an observation vector. As we impose stationary behavior on the top layer, each observation vector and its neighborhood would correspond to a local subspace in the top layer. Therefore although this hierarchical model is equivalent to a conventional linear dynamical system, the way the model is trained creates different parameters from the conventional model trained with the Kalman filter applied to the same data. This behavior was explained in [17].

TABLE II
The table shows performance of the state-of-the-art methods on pitch estimation of isolated notes.

YIN	SWIPE'	HLDS	Time Domain HLDS
93.71%	95.45%	96.61%	98.73%

In this paper we further detailed the properties of the model. We have shown its performance on music as well as isolated notes. We also gave an insight to the convergence time of the model and introduced the time domain implementation to overcome the limitations due to the slow convergence time. We have seen that the time domain implementation have outperformed the frequency domain implementation and consistently performed better then the state-of-the-art as seen in Table II.

This work is the first step in building a bigger model that can extract information on different time scales. With this initial building block we gather information about the note that is played and the transitions between notes. Our plan is to duplicate this same framework to work on the output of this layer. As we know the western music consists of a hierarchical structure that starts with a note, followed by a group of notes forming motifs, and different motifs tied together forming themes. We created the famous first motif in Beethoven's 5th Symphony using the trumpet notes we have. We repeatedly presented this motif to a trained model with different variations such as dynamical amplitude variations (crescendo or decrescendo) and tempo. We noticed that the same trajectory was followed in the state space of the highest layer. This behavior is presented in Figure 9. Our goal is to capture these hierarchical structures by forming further hierarchies using the HLDS framework. The model will be further investigated in the future regarding its generalization property, model size selection and will be extended to work on polyphonic input.

Due to our use of a linear system the computational burden

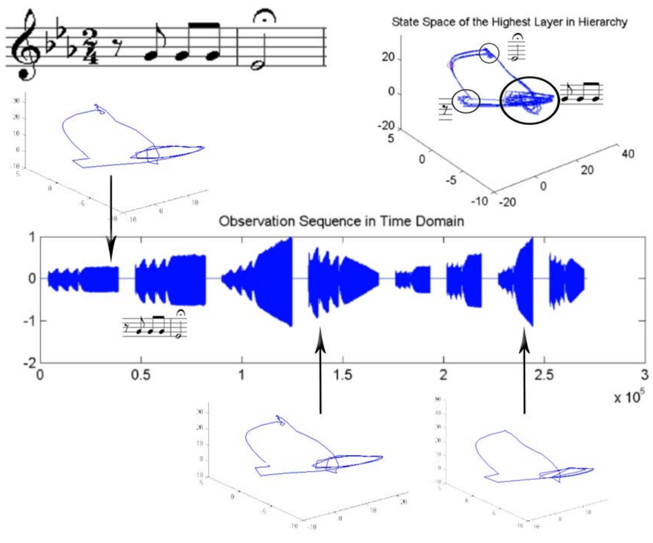

Fig. 9. The state trajectory when variations of the same motif is presented to the HLDS.

is reasonable. The algorithm takes ≈ 11.62 ms to estimate the states for each window (s=3, k=10, n=60) in the frequency domain implementation. This number grows linearly with the number of reiterations over the same window of data. Usually we reiterate 4 times over each window of new data thus the computation time for each window is ≈ 46.47 ms. As parameters are fixed after learning, the algorithm can work almost real time (each window contains ≈ 36.30 ms of new data). (The algorithm is tested in MATLAB R2011b, using Intel Core i5-2320 3.0GHz CPU with 6GB RAM)

References

[1] T Warren Liao, "Clustering of time series dataa survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[2] Stephen Handel, *Listening: An Introduction to the Perception of Auditory Events*, MIT Press, Cambridge, MA, 1993.

[3] L. Barrington, A.B. Chan, and G. Lanckriet, "Modeling music as a dynamic texture," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 602–612, 2010.

[4] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel, "The helmholtz machine," *Neural computation*, vol. 7, no. 5, pp. 889–904, 1995.

[5] Karl Friston, "A theory of cortical responses," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

[6] Rajesh PN Rao and Dana H Ballard, "Dynamic model of visual recognition predicts neural response properties in the visual cortex," *Neural Computation*, vol. 9, no. 4, pp. 721–763, 1997.

[7] Luke Barrington, Antoni B Chan, and Gert Lanckriet, "Dynamic texture models of music," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 1589–1592.

[8] Antoni B Chan and Nuno Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 5, pp. 909–926, 2008.

[9] Aapo Hyvärinen, Jarmo Hurri, and Patrik O Hoyer, *Natural image statistics*, vol. 39, Springer, 2009.

[10] Yonatan Vaizman, Roni Y Granot, and Gert Lanckriet, "Modeling dynamic patterns for emotional content in music," in *Proc. ISMIR*, 2011, pp. 747–752.

[11] Emanuele Coviello, Antoni B Chan, and Gert Lanckriet, "Time series models for semantic music annotation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1343–1359, 2011.

[12] John J Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[13] DW Tank and JJ Hopfield, "Neural computation by concentrating information in time," *Proceedings of the National Academy of Sciences*, vol. 84, no. 7, pp. 1896–1900, 1987.

[14] John J Hopfield, David W Tank, et al., "Computing with neural circuits-a model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[15] KP Unnikrishnan, John J Hopfield, and David W Tank, "Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections," *Signal Processing, IEEE Transactions on*, vol. 39, no. 3, pp. 698–713, 1991.

[16] Rungsun Munkong and Biing-Hwang Juang, "Auditory perception and cognition," *Signal Processing Magazine, IEEE*, vol. 25, no. 3, pp. 98–117, 2008.

[17] Goktug T. Cinar and Jose C. Principe, "Clustering of time series using a hierarchical linear dynamical system," in *ICASSP*, 2014.

[18] H.L. Read, J.A. Winer, and C.E. Schreiner, "Functional architecture of auditory cortex," *Current opinion in neurobiology*, vol. 12, no. 4, pp. 433–440, 2002.

[19] R.E. Kalman et al., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[20] Alex Nelson, *Nonlinear estimation and modeling of noisy time-series by dual Kalman filtering methods*, Ph.D. thesis, Oregon Graduate Institute of Science and Technology, 2000.

[21] V. Panuska, "A new form of the extended kalman filter for parameter estimation in linear systems with correlated noise," *Automatic Control, IEEE Transactions on*, vol. 25, no. 2, pp. 229–235, 1980.

[22] University of Iowa Electronic Music Studios , "Musical instrument samples," `http//:theremin.music.uiowa.edu/`, 1997, [Online; accessed 24-April-2012].

[23] *The Second Golden Beatles Album*, Hansen Publications, Inc., 1966.