ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-271

Groovy Neural Networks

Axel Tidemann[1] and **Yiannis Demiris**[2]

Abstract. The drum machine has been an important tool in music production for decades. However, its flawless way of playing drum patterns is often perceived as mechanical and rigid, far from the groove provided by a human drummer. This paper presents research towards enhancing the drum machine with learning capabilities. The drum machine learns user-specific variations (i.e. the groove) from human drummers, and stores the groove as attractors in Echo State Networks (ESNs). The ESNs are purely generative (i.e. not driven by an input signal) and the output is used by the drum machine to imitate the playing style of human drummers, making it a cost-effective way of achieving life-like drums.

1 INTRODUCTION

The research in this paper is aimed to enhance the cost-effective and easy way of creating drum tracks that is possible with current music production software (e.g. Logic, Pro Tools, Cubase) with the *groove* of a human drummer. The drum machine is a much cheaper alternative to recording live drums. However, the drum machine plays patterns without flaws, which makes it sound rigid and machine-like. Human drummers vary the way patterns are played, both on a small-scale level (varying dynamics and tempo) and on a large-scale level (changing the pattern to be played altogether, such as playing a break). These variations constitute the *groove* of the drummer. Current music production software have parameters that can be tweaked to achieve a human-like effect, however these variations add random noise with the intention that these variations will be perceived as human - the software has no understanding of what makes a drum pattern groovy. The research presented in this paper models these user-specific variations with recurrent neural networks, and demonstrates that the networks are able to represent both the small and large-scale variations that make a drummer groovy. These networks are then used to imitate the playing style of the drummers that served as teachers. The result is a groovy drum machine.

2 BACKGROUND

Modeling user-specific variations in playing style has been a field of study for years within the AI community. Saunders et al. [12] use string kernels to identify the playing style of pianists, by looking at changes in beat-level tempo and beat-level loudness. However, imitating the *style* of the pianists was not attempted. Tobudic and Widmer [15] also consider variations in tempo and dynamics as the two most important parameters of expressiveness. To learn the playing style of a pianist, they use first-order logic to describe how the pianist would play a certain classical piece, and then a clustering algorithm to group similar melody lines (i.e. phrases) together. They use the models to play back music in the style of given pianists. Pachet's Continuator uses Markov models to create a system that allows real-time interactions with musicians [8], however his focus is more on replicating the *tonal* signature of a musician; the Markov model represents the probabilities that a certain note will follow another. A musician plays a phrase and the Continuator will then play another phrase which is a *continuation* of that phrase. Mantaras and Arcos use case-based reasoning to generate expressive music performance by imitating certain expressive styles, such as joyful or sad [2]. Raphael [10] has implemented a real-time system for accompanying soloists, "Music Plus One". The system allows soloists to play along with an orchestra played by a computer, after the soloist has "practiced" along with the system. The idea is to model how a soloist tends to vary the tempo when playing a classical piece of music, making the orchestra (i.e. the computer) follow the soloist.

Current sophisticated drum sample software (e.g. FXpansion BFD, Toontrack EZdrummer, DigiDesign Strike, Reason Drum Kits, Native Instruments Battery) contains gigabytes of samples that closely match the acoustic response to playing dynamics. However, the drum libraries still need to be programmed, since they provide no intelligent way to generate human-like drum patterns. This must be done by the user, either by programming the pattern himself or choosing a rhythm template. The drum software contains parameters that can be tweaked to enhance the realism of the produced tracks (typically a "groove engine" where it is possible to increase randomization of beats and/or adjust timing and velocity), in addition to manually changing programmed patterns. Still, the user needs to know how to achieve the desired result, since the software has no understanding of how to generate human-like drum patterns. If the user could buy a "drummer in a box" that had a *model* of how a real drummer plays a certain pattern, it would greatly reduce the cost of having life-like drums. We believe this could be an important tool for musicians, since the programming of the drums would be easier and the user could select the drummer of his preference to perform on his tracks.

3 ARCHITECTURE

The architecture for learning and imitation of drum patterns is called "Software for Hierarchical Extraction and Imitation of drum patterns in a Learning Agent" (SHEILA), see figure 1. SHEILA learns drum patterns and the playing style of human drummers, and stores them in a library. After training, SHEILA can be used as a groovy artificial drummer, capable of imitating the playing style of the drummers that provided the drum patterns used as training data. The drumming domain is suited for time-dependent sequential modeling due to its repetitive nature, since the groove of a drummer manifests over time.

[1] IDI, NTNU, Norway, email: tidemann@idi.ntnu.no
[2] ISN, Imperial College London, UK, email: y.demiris@imperial.ac.uk

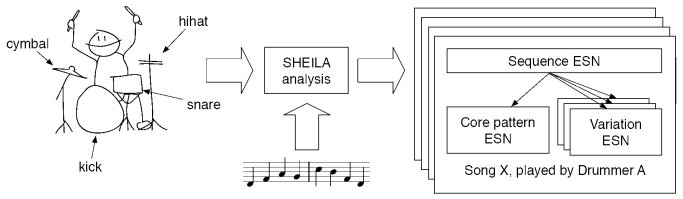

Figure 1. The SHEILA architecture. Playing style is learned in a hierarchical fashion by learning large-scale variations (i.e. variations of a pattern), as well as small-scale variations (variations of dynamics and tempo). All the models are represented by Echo State Networks (ESNs). The melody is used to group similar drum patterns together. After the training phase, SHEILA can imitate the playing style of the drummers that served as teachers.

3.1 Modeling the groove using neural networks

The inputs to SHEILA are drum patterns and the accompanying melody, both represented in the MIDI[3] format. SHEILA models both small- and large-scale variations of drum patterns using Echo State Networks (ESNs). An ESN is a recurrent neural network characterized by a large sparsely connected hidden layer, where only the output layer weights are modified during learning [6]. The input layer weights are generated at random, and left unchanged during training. By only modifying the output layer weights, training the network is reduced to a linear regression problem, which is a lot cheaper computationally compared to the backpropagation through time algorithm. In SHEILA, the input layer is not used. The ESNs learn the sequences by teacher forcing, i.e. by writing the desired sequence of output states into the output nodes during the training phase. The only inputs to the hidden layer comes from the output nodes. After training, the teacher forcing stops, and the network runs on its own. The ESN continues to generate the desired output sequence due to the reverberations of the hidden layer dynamics. In other words, the desired sequence of output states is stored as an attractor in the ESN. How this is used in SHEILA will be explained in the following sections. Why use ESNs to represent the groove sequence? Apart from the obvious advantage of using a dynamical system with memory capabilities, it also draws on biological inspiration; neuroscientific findings suggest specific areas of the brain process temporal musical information [11]. To model a human quality such as groove, using a technique that is modeled on how the brain works seems like a step in the right direction.

The name of the drummer and the song is also stored in the SHEILA library. Different drummers play the same pattern, but in their own style. The name of the drummer and the song will then help the user of SHEILA to decide which style he wants on the imitated drum track when SHEILA is used to imitate a drummer.

3.1.1 Large-scale variations

Large-scale variations are defined as changes in the actual pattern played, such as playing a break instead of a certain pattern. The most commonly played drum patterns are denoted *core patterns*. Core patterns are found by a recursive process: first the different parts of the melody are found (what in common music terms would be referred to as the verse/chorus/bridge of the song). This is achieved by transforming the melody into a string, and searching for *supermaximal repeats*, an approach used in computational biology to discover sequences of genes [4]. A supermaximal repeat is defined as a recurring pattern that is not a substring of any other pattern. Similar parts are grouped together, and the core patterns are the most commonly played drum pattern within the similar parts. To find the most commonly played drum patterns, the same search for supermaximal repeats is performed. Patterns correspond to one bar of the MIDI note sequence (i.e. 4 quarter notes). Patterns that differ from the core pattern within a melodic segment is considered to be a large-scale variation of the core pattern. For each song, there will be several core patterns, corresponding to the melodic segments. Core patterns are written as C_x, whereas variations are written as C_xV_y. From the low-level MIDI data a high-level representation of the song is found, namely the sequence of drum patterns. The sequence of patterns within a melodic segment is represented by an ESN (from now on referred to as ESN_{seq}). The string sequence is transformed into a binary matrix where one row corresponds to one bar, and the location of the high bit indicates which pattern (core or variation) to play. This sequence is then teacher forced to the ESN_{seq}, which produces the same output sequence after the training phase. The design choice to have one ESN_{seq} for similar melodic segments was made because it is the intention that SHEILA will later be used in a different setting, where only specific core patterns (and their variations) are to be played. If the ESN_{seq} learned the large-scale sequence of the entire song, it would only be suited to play back that particular song.

3.1.2 Small-scale variations

Small-scale variations are defined as variations in timing (i.e. how much the drummer is before or after the metronome) and dynamics (i.e. how hard a beat it played, also referred to as velocity) that occur when a drummer is playing a pattern. After defining the core patterns and variations, the similar drum patterns are grouped together, and the MIDI data is transformed into a target matrix where one row represents one timestep of the MIDI data. Quantizing the raw MIDI data allows for calculation of how much a note was before/after the metronome. The placements of the velocity and onset time data also code for which note was played (for instance, hihat, snare drum or kick drum). The velocity and onset times were scaled to the range [0,1]. One ESN represent the sequence of velocities (denoted ESN_{vel}), one ESN represent the sequence of onset times (ESN_{ons}). Early experiments tried to combine both onset time and velocity in one ESN, but finding a stable solution was difficult. Closer examination of the data revealed that the onset times had a variation that was on a slower timescale than that of the velocities. The onset times varies over several bars, whereas the velocity varies greatly from one note to the next (this will be elaborated in section 4). The spectral radius of the ESN describes the speed of the internal dynamics of the ESN, and is the most important parameter to tune [5]. It was therefore crucial that the velocity and onset times were represented on different networks, since this parameter needed to be different for each network. After the division was made, finding stable solutions became a lot easier.

The grouped patterns are then used to train ESNs that represent their variations in velocity and timing, resulting in specific ESNs for the core pattern and for each of the variations of the core pattern.

3.2 Imitating the groove

When the training is completed, the user of SHEILA presents a desired pattern in the MIDI format. Upon recognition, the user can choose which drummer should play the desired pattern. The name

[3] Musical Instrument Digital Interface, a protocol for electronic music equipment to communicate in real time.

of the drummer and which song the pattern was played on will aid the user to decide. The user then specifies how many bars the desired pattern should be played. SHEILA then runs the ESN_{seq} of the desired pattern for the desired number of bars, outputting the sequence of core patterns and variations. The corresponding ESN_{vel} and ESN_{ons} are then run for the desired number of bars; the output results in MIDI data. The ESN_{seq} introduces large-scale variations, and the ESN_{vel} and ESN_{ons} introduce small-scale variations. Recall that the ESNs are all purely generative, they are not driven by input at all. However, they need to be given a starting state, which is the last state of the hidden and output layer of the network during training.

4 EXPERIMENTAL SETUP

The SHEILA system was implemented in MatLab. To find the supermaximal repeats, *vmatch*[4] was used, which is implemented using the algorithms described in [1]. Propellerheads Reason 3.0[5] loaded with Reason Drum Kits was used for recording MIDI signals and for generating sound from MIDI files. Recording MIDI was done with a Roland TD-3[6] velocity sensitive electronic drum kit. Five male amateur drummers (average age 27.8) recorded drum tracks to a melody written by the authors, and were told to play specific patterns for the verse (shown in figure 2), chorus and bridge. At each 8th bar of the verse, there was a break that they had to play the same way. Apart from these directions, the drummers were free to introduce large-scale variations as they saw fit. The overall structure of the song was verse/chorus/verse/chorus/bridge/chorus/chorus. The tempo was 120 beats per minute (BPM), yielding the length of the song 2:30 minutes. The ESN_{seq} had 50 hidden nodes, and a spectral radius $\alpha = .99$. The spectral radius describes the speed of the internal dynamics of the ESN; $\alpha = .99$ specifies slow dynamics. This was chosen because the timescale of the ESN_{seq} were often long and highly irregular. The ESN_{vel} and ESN_{ons} on the other hand, represent faster dynamics since their role is to capture a short cycle in a long stream of data. By examining the training data, the timescale of which variations occur was discovered to be different on velocity and timing data. This can be seen in figure 3, which shows the velocity and timing data for the hihat sequence that correspond to the pattern in figure 2. Observe how the velocity data vary greatly from one note to the next, whereas the onset time varies more slowly.

To account for these observations, $\alpha = .1$ for the fast ESN_{vel}, $\alpha = .4$ for the slower ESN_{ons}. These values are found by experimentation, as recommended by Jaeger [5]. The networks started out with 10 nodes in the hidden layer. Finding a stable solution in a purely generative ESN was not guaranteed in every trained ESN. The training error would be low for every network, but once left to run on its own, some networks tended to oscillate in an unstable manner. To overcome this problem, each ESN was run for the same length as the training data, and if the resulting output pattern differed more than 10% from the training pattern, it was discarded and another ESN was created, trained and tested. If five consecutive ESNs had an error greater than 10%, the number of nodes in the hidden layer was increased by 1. In practice, this simple heuristic would guarantee that a solution would be found rather quickly, with less than 25 nodes in the hidden layer. Recall that the training of the ESN is a simple linear regression task; training and testing an ESN on some of the longer sequences (e.g. a 144 x 3 matrix) takes less than a second on an 1.8GHz iMac G5 running MatLab 7.

[4] www.vmatch.de
[5] www.propellerheads.se
[6] www.roland.com

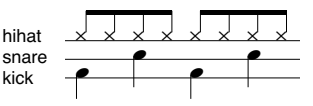

Figure 2. One of the three core patterns the drummers were required to play in the experiment.

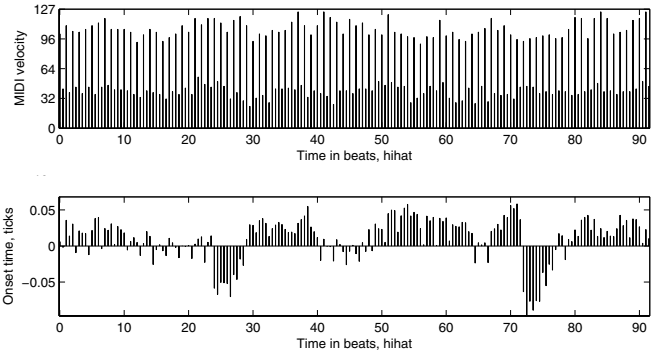

Figure 3. The plots show the training sequence of hihat velocity and onset time when drummer B played the pattern in figure 2. The difference from one velocity to the next is much larger than the difference from one onset time to the next, which fluctuates on a much slower timescale.

5 RESULTS

To evaluate the imitation performance of the SHEILA architecture, it was set to play back the same song structure used during training. Table 1 shows how many large-scale variations a drummer would introduce when playing a core pattern in addition to how often variations were played instead of the core pattern, calculated from the original training data. The table shows how some drummers tends to introduce many variations and play them often, whereas other drummers tend to play just the pattern they were told to play. This indicates the complexity of the sequence the ESN_{seq} had to learn, and the complexity of the imitated sequence.

Table 1. The tuples represent how many unique variations the drummers introduced when playing a core pattern, and how often variations in total were played instead of a core pattern (keeping in mind that a particular variation can be repeated throughout the sequence). This indicates the complexity of the sequence of large-scale variations and core patterns.

Drummer	A	B	C	D	E
C_1	5, 54%	3, 18%	7, 43%	2, 7%	5, 46%
C_2	10, 41%	2, 22%	8, 41%	1, 3%	11, 63%
C_3	6, 75%	2, 38%	5, 63%	0, 0%	5, 63%

The small-scale variations of both velocity and onset time can be modeled using a Gaussian distribution, as described in [13]. One of the leading software samplers on the market, FXpansion BFD, use the same approach to model human variations in its "Humanize panels"[7]. This allows for a simple way to show the small-scale variations present in both the original and imitated data. Figures 4 and 5 show how SHEILA models the small-scale variations of drummers A and E playing the pattern shown in figure 2 (due to space limits,

[7] See page 118 of the user manual (accessed 2008-05-26), www.fxpansion1.com/resourceUploads/BFD_Manual_English.pdf

the graphs for all drummers cannot be shown). Figure 4 shows how drummer A strongly accentuates (i.e. periodically varies the velocity of) the hihat beats, whereas drummer E has a more even velocity profile for the hihat beats.

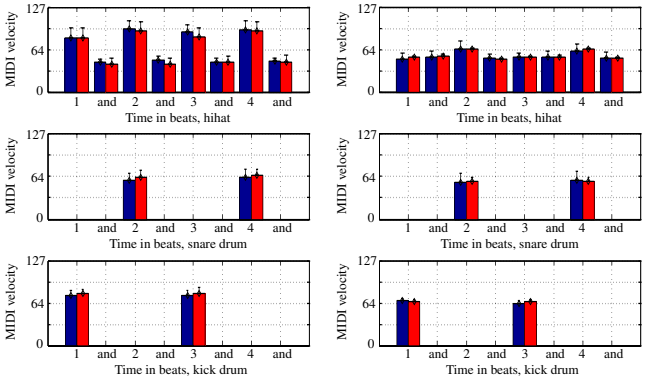

Figure 4. To the left is the velocity profile for drummer A, playing the pattern shown in figure 2. The Y scale is $[0-127]$, corresponding to the MIDI resolution. The X scale corresponds to the beats in the measure, which is a common way to count when playing music. The blue bar stems from the training data, the red bar from the output of SHEILA, when instructed to play the same song as that of the training input. The similarity between the blue and red bars indicate that the ESN_{vel} successfully captures the small-scale variations of the training data. Notice also how the velocity profile differs from that of drummer E (to the right). Most easily seen is how the accentuation (i.e. variation of velocity) on the hihat is not as pronounced as for drummer A; this is a manifestation of the different grooves of drummers A and E.

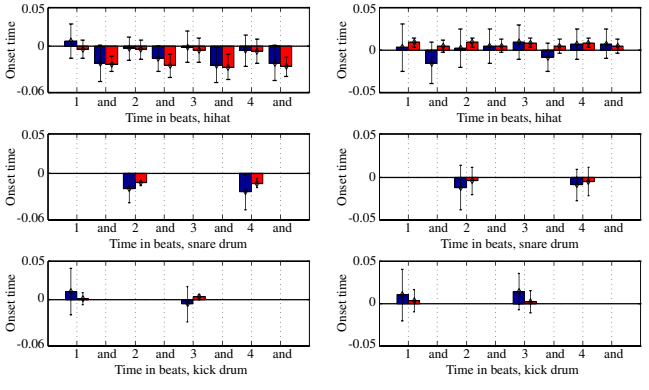

Figure 5. To the left is the onset time profile for drummer A, playing the pattern shown in figure 2. The Y scale is onset time in *ticks*. There are 120 ticks in the range $[0-0.99]$ between each quarter note. The X scale corresponds to the beats in the measure, similar to figure 4. As in figure 4, the blue bar is the statistics from the training data, the red bar is the analysis performed on the imitation done by SHEILA, showing that the output of the ESN_{ons} resembles that of the training data. The plot shows how drummer A tends to be ahead of the metronome when playing the pattern in figure 2. To the right is the onset time plot for drummer E. The onset times tend to be more centered around the metronome for the hihat beats, distinctively more than for drummer A, which contributes to the difference of groove between drummers A and E.

The onset time plays an important role in how aggressive/relaxed drum patterns are perceived, depending on whether the onset time is before or after the metronome. Figure 5 reveals that drummer A tends to be ahead of the metronome (yielding a more aggressive feel), whereas drummer E tends be more centered around the metronome, for a more "tight" feel. The authors are aware that these terms are vague but acoustically distinct; we encourage the reader to listen to available MP3 files[8] that better demonstrate these differences (included are imitations performed by SHEILA). Figures 4 and 5 show the mean and standard deviation for both velocity and onset time, both for the original data and the imitated output. The similarity between the plots shows how SHEILA successfully models the small-scale variations, in addition to demonstrating that drummers A and E plays the same pattern with a different groove.

To assess both the large- and small-scale differences between original and imitated drum tracks, as well as between drummers, a sequence similarity metric was implemented as described in [7]. The cost function was adapted to account for differences in velocity as well as timing of events, e.g. by adding the normalized difference in velocity between two events. The similarity metrics can be seen in table 2. The metrics show that imitations are similar to originals, and that the drummers have different styles when compared to another. The difference when comparing originals to imitations and drummers to each other is generally an order of magnitude. However, note that the metrics only have value as relative *comparisons* between the MIDI sequences. They do not represent an absolute difference. Yuijan and Bo have recently developed a *normalized* metric [16], however it does not account for timed series; this appears to be an open research issue, and beyond the scope of this paper. Still, the similarity metrics indicate a strong degree of similarity between original drum tracks and imitations (which is further backup up by figures 4-5), and that each drummer has a playing style different from the others.

Table 2. (a) shows the similarity metric described in [7] when comparing original drum tracks to SHEILA's imitations, (b) compares drummers to other drummers. The metrics indicate that the originals and imitated drum tracks are similar, and that the different drummers have different playing styles.

Original	A	B	C	D	E
Imitation	0.46408	0.37102	0.37176	0.60169	0.37995

(a)

	A	B	C	D	E
A	0	5.185	5.8272	6.1193	6.9911
B	5.185	0	5.4271	1.944	5.4166
C	5.8272	5.4271	0	6.0649	6.4713
D	6.1193	1.944	6.0649	0	6.135
E	6.9911	5.4166	6.4713	6.135	0

(b)

Another important aspect of the onset time is the tempo drift that occurs over time. A drummer will constantly be before or after the metronome, which will make the tempo fluctuate over time, as can be seen in figure 3. Figure 6 shows how the output of SHEILA induced the same drift in tempo over time as that of the original drum sequence. To examine how the ESN store the grooves as attractors, plots were made of hidden layer nodes during a run where the ESN was generating output. Figure 7 shows plots for some hidden nodes of the ESN_{vel} of the pattern in figure 2 for drummer A. The ESN_{vel} was run for 240 timesteps (double what it was trained on). The figures show that the activation patterns have stable attractor shapes, but with deviations. This is a further testament to how small-scale variations are introduced when imitating a certain pattern; these deviations will make the output slightly different over time. But since

[8] www.idi.ntnu.no/∼tidemann/sheila

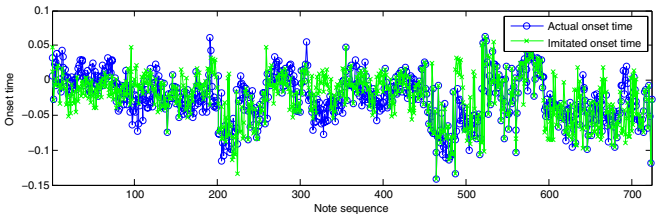

Figure 6. Tempo drift throughout the song, drummer A. The circle plots show the tempo drift present in the recorded drum patterns. The cross plots show the onset times during imitation. Observe how both the original and imitated note sequence drift over time in a similar fashion.

the attractors are modeled from the patterns from a human drummer, the fluctuations will be *similar* to that of the human drummer.

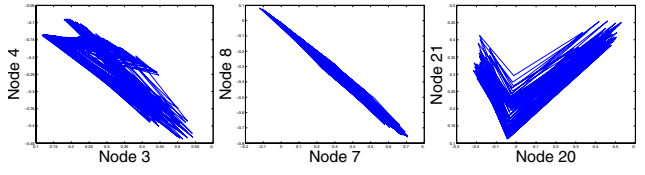

Figure 7. Attractor plots, for some randomly selected hidden layer nodes of the ESN_{vel} of the pattern in figure 2, drummer A. The ESN_{vel} was run for 240 timesteps (twice the training length). The plots are stable, but with deviations implying that the output will be slightly different over time.

6 DISCUSSION AND CONCLUSION

The choice of using MIDI was based on simplicity regarding the gathering and analysis of data. Another advantage with MIDI is that it will disregard the sound of the drums, which often will help to identify a drummer. The MIDI data allows focusing on the playing style of the drummer, which is the aim for our research.

By using ESNs, SHEILA is able to model the human quality that is *groove* by using a biologically inspired computational model. The model encompasses the quality of varying the output like that of a human drummer, making the output different from the original but still *recognizable*. The research presented in this paper enables the drum machine to become closer to a groovy human drummer. Effectively, the results will make it cheaper and easier to create human-like drum tracks when making music.

7 FUTURE WORK

SHEILA depends on MIDI information gathered using a MIDI drum kit. Acquiring the data is expensive; extracting the drum patterns and melody line directly from sound files would give access to vast amounts of training data; possible approaches are described in [9]. Apart from ease of computation, the reason for recording live drummers was an interest in making SHEILA learn the *physical* playing style of human drummers, i.e. the movement of the arms, upper torso and head. This would be used to visualize SHEILA. During the experiments conducted for this paper, motion tracking was also done. The goal is to be able to use SHEILA in a live setting as an *accompanying musician*, interacting with humans playing other instruments. This work will continue research done with motion tracking and subsequent imitation of arm movements as described in [14], using multiple forward and inverse models as building blocks for a motor control architecture [3].

ACKNOWLEDGEMENTS

The authors would like to thank the referees who helped improve the paper, and the drummers who participated in the experiment (Daniel Erland, Inge Hanshus, Sven-Arne Skarvik, Tony André Søndbø).

REFERENCES

[1] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch, 'Replacing Suffix Trees with Enhanced Suffix Arrays', *Journal of Discrete Algorithms*, **2**, 53–86, (2004).
[2] Ramon Lopez de Mantaras and Josep Lluis Arcos, 'AI and music from composition to expressive performance', *AI Mag.*, **23**(3), 43–57, (2002).
[3] Yiannis Demiris and Bassam Khadhouri, 'Hierarchical attentive multiple models for execution and recognition of actions', *Robotics and Autonomous Systems*, **54**, 361–369, (2006).
[4] Dan Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press, New York, NY, USA, 1997.
[5] Herbert Jaeger, 'Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network"', Technical report, German National Research Institute for Information Technology, (2005).
[6] Herbert Jaeger and Harald Haas, 'Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication', *Science*, **304**(5667), 78–80, (2004).
[7] H. Mannila and P. Ronkainen, 'Similarity of event sequences', *TIME*, 136–139, (1997).
[8] Francois Pachet, *Enhancing Individual Creativity with Interactive Musical Reflective Systems*, Psychology Press, 2006.
[9] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gomez, S. Streich, and B. Ong, 'Melody transcription from music audio: Approaches and evaluation', *IEEE Transactions on Audio, Speech and Language Processing*, **15**(4), 1247–1256, (May 2007).
[10] Christopher Raphael, 'Orchestra in a box: A system for real-time musical accompaniment', in *IJCAI workshop program APP-5*, pp. 5–10, (2003).
[11] Séverine Samson and Nathalie Ehrlé, *The cognitive neuroscience of music*, chapter Cerebral substrates for musical temporal processes, Oxford University Press, 2004.
[12] Craig Saunders, David R. Hardoon, John Shawe-Taylor, and Gerhard Widmer, 'Using string kernels to identify famous performers from their playing style.', in *ECML*, eds., Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, volume 3201 of *Lecture Notes in Computer Science*, pp. 384–395. Springer, (2004).
[13] Axel Tidemann and Yiannis Demiris, 'Imitating the groove: Making drum machines more human', in *Proceedings of the AISB symposium on imitation in animals and artifacts*, eds., Patrick Olivier and Chris Kay, pp. 232–240, Newcastle, UK, (April 2007).
[14] Axel Tidemann and Pinar Öztürk, 'Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA', in *IEA/AIE 2007*, volume 4570 of *Lecture Notes in Computer Science*, pp. 291–302. Springer Verlag, (June 2007).
[15] Asmir Tobudic and Gerhard Widmer, 'Learning to play like the great pianists.', in *IJCAI*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 871–876. Professional Book Center, (2005).
[16] Li Yujian and Liu Bo, 'A normalized levenshtein distance metric', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(6), 1091–1095, (2007).