Realtime Sample Selection Based Upon Timbral Similarity: A Comparison Between Two Methods

Arne Eigenfeldt
School for the Contemporary Arts
Simon Fraser University
Burnaby, Canada
Telephone number, incl. country code
arne_e@sfu.ca

Philippe Pasquier
School of Interactive Arts and Technology
Simon Fraser University
Burnaby, Canada
Telephone number, incl. country code
pasquier@sfu.ca

ABSTRACT

A comparison is made between two systems of realtime sample selection using timbral proximity that has relevance for live performance and/or game design. Samples in large sample libraries are analysed for audio features (rms, spectral centroid, energy, flatness, and spectrum using a Bark auditory modeler), and this data is statistically analysed and stored. Two methods of organization are described: the first uses fuzzy logic to rate sample similarity, the second uses a self-organizing map. The benefits and detriments of each method are described.

Categories and Subject Descriptors
J.5 [**Computer Applications**]: Arts and Humanities – *music.*

General Terms
Performance, Design.

Keywords
Timbre, realtime systems, heuristics, self-organising maps.

1. INTRODUCTION

Performers involved in realtime electroacoustic music are often faced with choosing their sounds during performance. It is not unusual for these performers to have hundreds, if not thousands, of available samples from which to choose. Lacking the ability to audition audiofiles during performance, the performer is forced to remember the subtle difference between "acoustic bass 2" and "acoustic bass 2a". Similarly, sound design for games may involve situations in which an initial sample needs to be followed with a similar timbre.

Metadata is one solution to this problem, through the inclusion of a description of the sound within the file itself. There are two problems with metadata: firstly, the reliance is upon the description to be accurate and complete; secondly, long descriptions are not useful in performance, where instantaneous decisions must be made regarding which sample to choose. Therefore, we propose that techniques from machine learning can be used to help make the decision.

A sample library is analysed, prior to performance, for a variety of features, including peak and RMS amplitude, brightness (spectral centroid), loudness (spectral energy), noisiness (spectral flatness), and spectrum (using the Bark scale auditory model spectrum analysis). Each sample's analysis data is indexed by filename. In performance, this file is recalled, and its data can be interpreted in different ways, depending upon the needs of the performer. Two methods of organization are described: the first uses fuzzy logic operators to rate sample similarity; the second uses a self-organizing map (SOM).

Section 2 describes related work in realtime timbre analysis, and discusses how our research advances the state of the art; Section 3 describes the analysis and preparation of the database prior to performance for both methods; Section 4 describes how the heuristic method is used, and how it can be used in performance; Section 5 describes how the self-organizing map is created, and how it can be used in performance; Section 6 compares the two methods; Section 7 offers our conclusions and future directions.

2. RELATED WORK

2.1 Realtime Timbre Recognition

Timbre is finally available as a control structure for realtime music. Early work in this area was done by Lippe [1], who used Max and the IRCAM Signal Processing Workstation (ISPW) to analyse timbre in performance.

More recently, Hsu [2] used realtime timbre recognition of saxophone to guide an interactive system. Ciufo [3] used Jehan's MSP external analyzer~ [4] to analyse incoming audio, which in turn influenced processing. Similarly, Rebelo and Renaud [5] used the same object in the BLISS laptop improvisation ensemble to analyse ensemble member's spectral data, information that is shared amongst the ensemble. Jehan's MSP external analyzer~, as well as other realtime MSP analysis tools, are allowing composers to explore the potential of timbre as a control element during performance.

The recent appearance of Music Information Retrieval (MIR) algorithms in ChucK [6] will, no doubt, precipitate many new realtime works that involve timbral recognition, previously only possible using non-realtime tools.

2.2 Timbre Organisation

The Music Information Retrieval (MIR) community has done considerable research into timbral organization, specifically in determining similarity between songs [7].

Cano and Koppenberger [8] describe a system which classifies a huge sound effects database discovering timbral similarities between sounds, and labeling these sounds with similar tags. Pampalk, Dixon, and Widmer [9] describe a system with combines descriptors derived from audio analysis with meta-information to create different views of a music collection, using an aligned self-organising map. This builds upon work by Logan [10] who uses a spectrum-based similarity measure to create playlists of similar songs, and Auccounturier and Pachet [11], who use a spectrum-based similarity measure to find similarities between songs. Lübbers [12] describes a system called "The Sonic SOM", which uses a SOM that does not function in realtime, to help users understand music collections. Knees et al. present another system [13] that uses text-retrieval methods to find meta-data extracted from the Web in order to discover song similarities.

2.3 Differences from previous work

The work described here is focusing upon realtime selection of samples based on their timbral properties, and is thus different from most previous timbral analysis research. Furthermore, it does not use meta-data to organize databases, allowing it more flexibility.

As the authors are both composers, the intentions are also markedly different than MIR research: rather than attempting to navigate a search space and return an exact match, the interest is in similar timbres, rather than specific matches. For example, in cases where the software is used to respond to live performer input in order to select similar timbres, the system response should not necessarily be limited to exact sample matches (i.e. timbale samples for timbale input), but timbres that have "similar" (or related) spectral content.

While the entire system is not realtime, in that the sample database must initially be analysed (see Section 3.2) and, in the case of method 2, the SOM must be trained (see Section 5.2), the final product was created for performance use.

Lastly, the methods described here were coded in Max/MSP, and are intended to present useable tools for composers and performers, for use in performance. Comparing two different methods will hopefully allow others to incorporate these methods, or adjust them to suit their own needs

3. SAMPLE ANALYSIS

3.1 Sample libraries

Three different sample libraries were used in this research: a library of 1551 individual percussion samples; 379 Apple GarageBand loops, including pitched instrumental loops as well as unpitched percussion loops; 83 soundscape recordings, averaging approximately thirty seconds in duration.

3.2 Sample Analysis

Analysis of individual soundfiles was done using a MaxMSP patch using Jehan's analyzer~ object to derive the following feature data for each sample:
- brightness (spectral centroid)
- loudness (spectral energy)
- noisiness (spectral flatness)
- 24 band Bark analysis

Additionally, peak amplitude and rms analysis was done using the standard MSP objects peakamp~ and average~, while a frequency analysis was made using fiddle~.

Statistical analysis was done on this data over the course of the sample's duration, in order to determine the maximum, mean, and standard deviation for each feature.

Feature data is stored in the following format:

[Sample name] [duration in ms] [feature_max] [feature_mean] [feature_stddev] …

The maximum, mean, and standard deviation for the three highest Bark bands and their amplitudes are stored (see Table 1).

Each sample library's complete analysis file is written to disk as text, and is read prior to performance.

3.2.1 Bark Analysis

The Bark analysis [14] is auditory modeler that provides perceptually meaningful data, corresponding to the intensity of the first 24 critical bands of hearing. Furthermore, when compared to standard FFTs, the analysis itself already provides useful data reduction.

Table 1. Maximum Bark amplitudes for the three most intense bands of Asian Gamelan 04.

Band	Mean Amplitude
17	0.892
16	0.867
7	0.789

4. METHOD 1 DESCRIPTION

The first method for timbral selection is based upon a notion of spectral peaks within the Bark analysis. Each sample's three highest Bark bands - determined through maximum amplitude - are stored, and similarity is based upon the closeness of any two sample's Bark bands to one another. Fuzzy logic comparison operators [15] are used to determine closeness.

This method has been tested in an existing realtime generative rhythm composition system [16], and is more fully described elsewhere [17]. It works well for short percussion samples, since it does not take into account time-varying spectra.

As well as the analysis file described in Section 3.2, Method 1 requires two additional arrays to be created prior to performance.

An indexed array is created (sample_DB) that points to individual sample paths. A second array (bands_by_sample) is also created at this time, in which pointers to sample_DB indices are sorted by Bark energy bands.

For example, if a sample's Bark bands are (2 5 7), the sample number is entered into bands_by_sample at all three indices. This allows access to all samples that have high energy in a specific band.

4.1 Realtime Implementation

In performance, an input vector representing three Bark bands is passed to bands_by_sample, the array consisting of pointers to samples that contain specific bands. Thus, given an input vector of

(3 4 6), all samples whose three highest bands contain any of those values, is returned. The actual bands for each sample in this list is then compared to the input vector, and the number of direct matches is used to sort the list.

This algorithm works only if the input vector matches that of samples within the database, which is not always the case. Furthermore, only exact matches are returned, rather than similar matches. For this reason, fuzzy comparisons are made, by including adjacent bands in the search.

4.1.1 *Fuzzy logic comparisons*

A "fuzzy list" is created around the input vector by including adjacent bands: for example, given an input vector of (5 9 14), the following fuzzy list is generated: (4 6 8 10 13 15). Direct matches are summed; matches to the fuzzy list are summed and scaled by 0.66 (a hand-tuned value that created the ordering in Table 2) and the result normalized to be between 0 and 1 to create a closeness rating (see Table 2).

Table 2. Closeness ratings based upon matching bands

Direct matches	Fuzzy matches	Score
3	0	1.0
2	1	0.89
1	2	0.77
2	0	0.67
0	3	0.66
1	1	0.55
1	0	0.33
0	2	0.44
0	1	0.22

The inclusion of adjacent bands increases the number of samples to test. Incrementing through the list, in order, and selecting samples until the desired number is reached, would consistently create the same results, given the same input vector. In the case of the generative rhythm software, this is not a desired compositional choice, where, instead, different results are returned given the same input values. Therefore, the sample list is incremented randomly, until the desired number of selections is made.

4.2 Realtime applications

One possible application for this method is the selection of samples that are similar (or dissimilar) to timbres performed live. Incoming percussion timbres can be analysed to derive the 3 most intense Bark bands, and this input vector used to derive a set of similar samples.

Another application (used in the generative rhythm software) would be to select an approximate spectral bandwidth, from which the software can use Gaussian probabilities to generate a random input vector (see Figure 3).

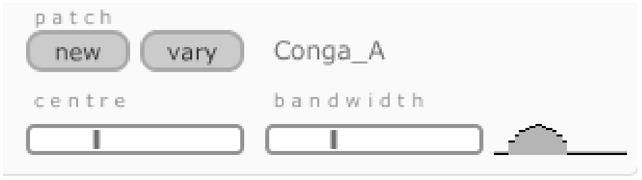

Figure 3. Defining a spectral band from which to choose a sample.

4.3 Search Heuristics

Since the search space can contain several hundred items, even when limited by using the pre-sorted bands_by_sample array, it was found that evaluating it in its entirety (using a 16-nearest neighbor implementation) took too long in performance, often several seconds. Therefore, a heuristic algorithm was created to find enough (16) acceptable solutions within an acceptable amount of time (less than 1500 ms).

When the randomly incremented search first begins, only those ratings above 0.66 are acceptable; after 250 ms, this is enlarged to include samples above 0.65, and after 1000 ms, it is enlarged to include 0.55 and above. During the search, results are added to the samples list and loaded, immediately available for performance.

4.4 Dissimilarity

Choosing dissimilar timbres to a given Bark set is simply a matter of creating an inverse probability vector around the three bands, then choosing three new bands using quantile probabilities from this vector, and finally searching for timbres with the new bands.

If the inverse selected bands are derived only once prior to selection, the resulting timbres will all be similar (since they share the same bands); however, the same inverse probability vector can generate a different set after each sample selection. This would result in each sample being dissimilar to the original, yet with a likelihood of dissimilarity within the new sample groups.

5. METHOD 2 DESCRIPTION

5.1 Self-Organizing Map

The second method for timbral selection uses a self-organizing map, or SOM [18]. Analysis data is clustered automatically by the SOM, according to similarities. Although any group of features can be used to compute similarities, the strength of this method relies upon the 2D visualisation of the resulting organisation (see Section 5.4).

5.2 Generation of the Self-Organizing Maps

A self-organizing map was implemented in MaxMSP to create a two dimensional representation of the sample database. SOMs are a type of artificial neural network using a neighborhood function so as to reflect feature proximity in a topologic manner.

For purposes of comparison to method 1, the input vector consisted of a sample's 3 highest Bark bands' numbers, between 0 and 23; however, the input vector can easily be changed to use any feature data, and be of any size (see Section 5.4). The visualization map was a 25 by 25 Jitter matrix, representing the 625 individual nodes (neurons), with the 3 weights of each node (one per band) being mapped to RBG values. The grid itself is toroidal, in that opposite edges are connected.

5.2.1 *Training*

Each node's weights are initialized to random values between 0 and 1. The number of training iterations (*ti*) was defined as 10,000 for the sample database of 1,551 items; the initial neighborhood size (*ins*) was half the size of the map: 12 nodes, in every direction. The initial learning rate was set to 1., and decreased at a logarithmic rate at each training iteration (*t*) to produce the learning rate (*lr*) using the following function:

$$\mathrm{lr}\,(t) = \log_{10}(t / ti) \qquad (1)$$

Similarly, the neighborhood size (*ns*) decreased at a logarithmic rate:

$$ns\,(t) = \log_{100}(\,t\,/\,ins) \qquad (2)$$

At each iteration, a training example, chosen randomly from the database, is fed to the network, and the Euclidean distance to all weight vectors computed. The winning vector, (the best matching unit, or BMU) is the node with the weight vector most similar to the vector of the input instance.

The weights of the BMU and those nodes in the neighborhood are then adjusted towards the input vector. The differences between their current weight and the input vector are multiplied by the weight scaling function (*ws*), and then added to the original weights:

$$ws = lr\,/\,(ns^2 + 1) \qquad (3)$$

Self-organisation initially takes place on the global scale (since the initial neighborhood is the entire map), whereas over time, the neighborhood shrinks to what is eventually a single node, and the weights converge to local estimates.

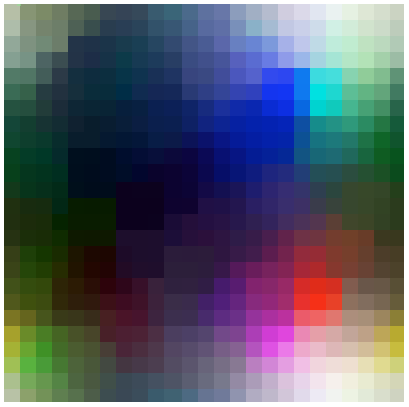

Figure 1. An example visualization of the SOM of the percussion sample library.

Once training is complete, the SOM will display those samples that are similar to one another in close proximity (see Figure 1). The actual colours displayed relate to the input vector (the 3 most intense Bark bands): in this case, the dark colours represent those samples in which all three Bark bands are low; the light colours represent those samples in which all three Bark bands are high; those that are closer to one primary colour represent samples whose Bark bands are spread out - i.e. a given Bark band vector of (17 3 5) will be bright red.

5.2.2 Mapping

Once training is complete, it is possible to directly associate individual nodes and input vectors. The database was incremented through in its entirety (one additional epoch), and fed to the SOM, with each BMU's sample number being stored in a separate array.

For the SOM shown in Figure 1, only 52% of the nodes had direct associations; however, of those with associations, the mean number of associations was 4.85, with a maximum of 139 sample files per node. This duplication of associations resulted from samples within the database having essentially identical data (i.e. the 3 most intense Bark bands). See Section 5.4 for a more productive use of the SOM.

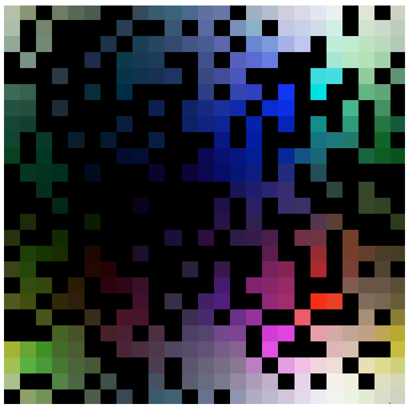

Figure 2. The associated SOM from figure 1, displaying the unassociated neurons as black.

The SOM can also be visualized to only show the direct associations, with non-associated nodes remaining black; thus, this visualization shows potential clustering of data (see Figure 2). Note, for example, the relative isolation of the two bright red squares in the bottom right, which signifies few similarities within the database to those samples. Compare this to the clusters of white squares at the extreme bottom right (which also wrap around to the top of the map): this signifies a clustering of similar samples within the database.

5.3 Realtime Implementation

An obvious benefit of using a SOM is its ability to visualize similarities within a database. The most practical realtime implementation for a SOM is to allow the user to click on the visual map, and derive the appropriate association(s) from the database. This is a simple matter of converting the <x,y> mouse selection coordinates into an index into the association array. Using the post-learning associative matrix (Figure 2), which clearly displays learnt associations, only those items in the database that directly correlate to the visual display will be returned.

5.4 Choice of features to display

As with any SOM, the effectiveness of the visualization is in the choice of features to display. It was found that the best features to choose were those that had significant differences within the database.

For example, Figure 4 displays the soundscape database, using the standard deviation over the sample's duration, of brightness, loudness, and noise. Thus, those squares that display more red are associated with samples that have more dynamic frequency (spectral centroid) change; those squares that display more green have more variation in their spectral energy; those squares that display more blue have more dynamic change in their spectral flatness.

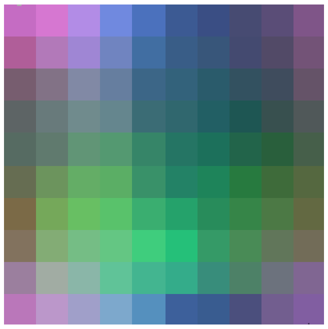

Figure 4. SOM visualizing the soundscape database, using standard deviation of brightness, loudness, and noise, after three epochs. Since the database has fewer items, the matrix was reduced to 10 x 10.

Figure 5 displays the selected GarageBand loop database, using mean brightness, standard deviation of brightness, and mean frequency. In this case, those squares that are dark contain low frequencies, and have little change in their spectral centroid (i.e. the bass samples); those squares that are yellow have higher spectral energy, with dynamic spectral change (i.e. the percussion samples).

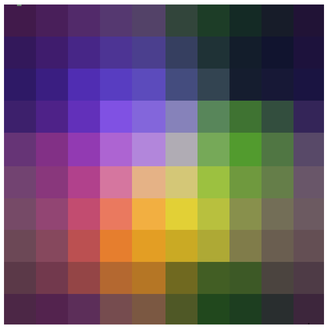

Figure 5. SOM visualizing the GarageBand loop database, using mean and standard deviation for brightness, and mean frequency.

5.5 Larger feature sets

Since the Bark analysis produces 24 discrete intensity values, it was possible to train the SOM using all 24 values as an input vector. Using the GarageBand loops, after 64 epochs (100,000 training iterations), the finer delineations between datum allowed more associations to be made. In the case of Figure 6, while only 46% of the squares had direct sample associations after 1 epoch, over 82% had associations after 64 epochs (see Fig. 6).

Visualizing 24 inputs in 2 dimensions required mapping the 24 weights of each node to the three RGB values. The averages over 8 bands' intervals were used. As a result, red squares indicate energy primarily in low bands; greens indicate energy in the midrange bands, and blues indicate energy in the high bands.

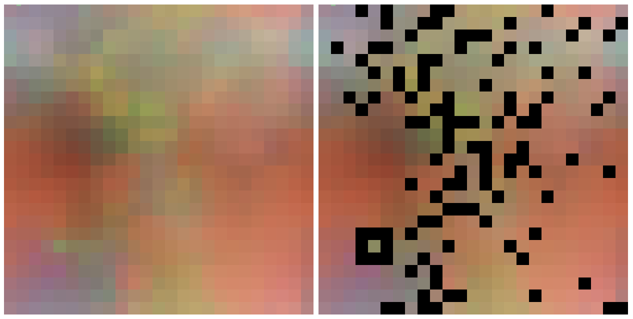

Figure 6. SOM visualizing 24 Bark bands of GarageBand loops, using mean amplitude (left) and the associative SOM, displaying black for non-associated nodes (right)

5.6 Realtime applications

If was found that the most effective use of SOMs in performance was to display several visualizations at once, allowing the user to make selections in one or more of the maps, then correlating the resulting selections. For example, using one SOM to display temporal information (such as Figure 4), and one to display frequency content, it is possible to select those items from the database that are associated with both selections.

Another useful application was to draw paths through the visualized data, which could be interpreted over time. For example, using the visualization presented in Figure 4, a path was drawn from the top left through the bottom middle, which was interpreted over a ten minute performance. During this time, samples could be automatically selected, beginning with dynamic frequency change, and ending with those with dynamic noisiness.

6. COMPARISON

It was found that both methods were useful, each offering benefits, and detriments, that need to be considered for the specific performance requirements.

6.1 Benefits to Method 1

Because the fuzzy logic system uses heuristics, enough acceptable solutions will always be found. In cases where a specific number of solutions is required (i.e. agent-based performers), this is of great benefit.

6.2 Detriments to Method 1

Although acceptable solutions will always be found, depending upon the input criteria, the level of acceptability can vary greatly. Since this method does not visualize the database, it is possible to define criteria which does not exist within the database. This is especially true if the database is relatively small.

6.3 Benefits to Method 2

SOMs visualize the actual database, which can influence the design of the database itself; for example, if not enough high frequency samples are present, the SOM can display this, and adjustments can be made.

Multiple SOMs can be used to display different features, and different features can even be visualized within a single SOM. In fact, SOMs are extremely flexible in their ability to display similarities over any kind of data.

Perhaps the greatest benefit of SOMs to realtime performers is the fact that no actual calculations - other than conversion of mouse location to database indices - is necessary: all the "heavy-lifting" is "front-loaded".

6.4 Detriments to Method 2

Although SOMs allow easy visualization, it is not immediately clear what is being visualized; for example, knowing what a red pixel represents, as opposed to a blue, requires an understanding of the analysis itself.

Since SOMs involve time-consuming training, they are not realtime, and so cannot display any new data.

Lastly, there is no easy way (at least compared to point and click) of selecting a range of selections or associations. In other words, the SOM (as discussed here) does not display the number of associations behind any square, and thus guaranteeing a certain number of selections (see Section 6.1) is more difficult.

7. CONCLUSIONS

We have described two different methods for realtime sample selection based upon timbral properties and similarity within large databases of samples. Both methods offer different benefits to the laptop performer, and both methods can be implemented within MaxMSP.

Two future directions are being currently investigated. In the first case, adding new samples to the database requires their analysis, and, in the case of method 2, retraining the SOM. This retraining could be overcome by borrowing node locations from similarly rated samples. In the second case, clustering algorithms could be incorporated within the SOM so as to position the associated nodes closer together and thus generate Islands of similarity [9].

This software, along with the mentioned generative rhythm software, was created in Max/MSP, and is available at the first author's website: [www.sfu.ca/~eigenfeldt/research.html].

8. REFERENCES

[1] Lippe, C. 1993. "A Composition for Clarinet and Realtime Signal Processing: Using Max on the IRCAM Signal Processing Workstation", in *Proceedings of the 10th Italian Colloquium on Computer Music,* Milan, Italy.

[2] Hsu, W. 2006. "Managing Gesture and Timbre for Analysis and Instrument Control in an Interactive Environment", in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Paris.

[3] Ciufo, T. 2005. "Beginner's mind: an environment for sonic improvisation", in *Proceedings of the International Computer Music Conference (ICMC),* Barcelona.

[4] Jehan, T., Schoner, B. 2001. "An Audio-Driven Perceptually Meaningful Timbre Synthesizer", in *Proceedings of ICMC.*

[5] Rebelo, P., Renaud, A. 2006. "The Frequencyliator – Distributing Structures for Networked Laptop Improvisation", in *Proceedings of NIME.*

[6] Fiebrink, R. et. al 2008. "Support for MIR Prototyping and Realtime applications in the ChucK Programming Language", in *Proceedings of the International Conference on Music Information Retrieval (ISMIR).*

[7] Tzanetakis, G., Cook, P. "MARSYAS: A Framework for Audio Analysis" in *Organized Sound, Cambridge University Press* 4(3), 2000.

[8] Cano, P., Koppenberger, M. 2004. "Automatic sound annotation", in *IEEE Workshop on Machine Learning for Signal Processing*, pages 391–400.

[9] Pampalk, E., Dixon, S., Widmer, G. 2004. "Exploring Music Collections by Browsing Different Views", in *Proceedings of ISMIR.*

[10] Logan, B. 2002. "Content-based playlist generation: Exploratory experiments", in *Proceedings of ISMIR.*

[11] Aucouturier, J.-J., Pachet, F. 2002. "Music Similarity Measures: What's the Use?", in *Proceedings of ISMIR.*

[12] Lübbers, D. 2005. "Sonixplorer: Combining Visualization and Auralization for Content-based Exploration of Music Collections", in *Proceedings of ISMIR.*

[13] Knees, P., Pohle, T., Schedl, M., Widmer, G. 2002. "Automatically describing music on a map", in *Workshop on Learning the Semantics of Audio Signals.*

[14] Jehan, T. 2005. *Creating Music by Listening.* PhD thesis, MIT Media Lab, Cambridge, MA.

[15] Klir, G. J., Yuan, B. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ.

[16] Eigenfeldt, A., Kapur, A. 2008. "An Agent-based System for Robotic Musical Performance", in *Proceedings of NIME.*

[17] Eigenfeldt, A., Pasquier, P. 2009. "Realtime Selectin of Percussion Samples Through Timbral Similarity in Max/MSP", in *Proceedings of ICMC.*

[18] Kohonen, T. 1984. *Self-organization and Associative Memory*, 3rd ed., Springer, New York.