Real-Time Timbral Organisation: Selecting samples based upon similarity[1]

ARNE EIGENFELDT* and PHILIPPE PASQUIER**

*School for the Contemporary Arts/ **School of Interactive Arts and Technology, Simon Fraser University, Burnaby, Canada
E-mail: *arne_e@sfu.ca; **pasquier@sfu.ca

A comparison is made between two systems of real-time sample selection using timbral proximity that has relevance for live performance. Sound files in large sample libraries are analysed for audio features (amplitude RMS, spectral centroid, spectral flatness, and spectral energy using a Bark auditory modeller), and this data is statistically analysed and stored. Two methods of organisation are described: the first uses fuzzy logic to rate sample similarity, the second uses a self-organising map. The benefits and detriments of each method are described.

1. INTRODUCTION

Performers involved in real-time electroacoustic music are often faced with selecting 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, for example, 'train station 1' and 'train station 1a'.

Metadata is one solution to the problem of real-time sample identification, through the inclusion of a description of the sound within the file itself. There are two drawbacks to using metadata: firstly, there is a reliance upon the description being 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 be used to help make such decisions.

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 (Zwicker 1961). 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 organisation are described: the first is a heuristic method that uses fuzzy logic operators to rate sample similarity, the second uses a self-organising map (SOM).

[1]This research was funded by a grant from the Social Science and Humanities Research Council of Canada.

Section 2 describes related work in real-time timbre analysis and discusses how our research to cluster samples by similarity 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 SOM 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. Real-time timbre recognition

Timbre is now available as a control structure for real-time electroacoustic music. Early work in this area was done by Lippe (1993) using Max and the IRCAM Signal Processing Workstation (ISPW) to analyse timbre in performance.

More recently, Hsu (2006) used real-time timbre recognition of the saxophone to guide an interactive system. Ciufo (2005) used Jehan and Schoner (2001) to analyse incoming audio, which in turn influenced processing. Similarly, Rebelo and Renaud (2006) used the same software in the BLISS laptop improvisation ensemble to analyse ensemble members' spectral data, providing information that was shared amongst the ensemble. These open-source real-time analysis tools written for Max/MSP allow composers to explore the potential of timbre as a control element during performance within a widely used software environment. Furthermore, the addition of the Small Music Information Retrieval Toolkit (Fiebrink, Wang and Cook 2008) to ChucK will, no doubt, precipitate many new real-time works that involve timbral recognition, previously only possible using non-real-time tools.

2.2. Timbre organisation

The Music Information Retrieval (MIR) community has done considerable research into timbral organisation, specifically in determining similarity between songs (Tzanetakis and Cook 2000). Cano and Koppenberger

Organised Sound **15**(2): 159–166 © Cambridge University Press, 2010. doi:10.1017/S1355771810000154

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

(2004) describe a system that classifies a huge sound-effects database, discovers timbral similarities between sounds, and labels these sounds with meta-data tags. Pampalk, Dixon and Widmer (2004) describe a system which combines descriptors derived from audio analysis with meta-information to create different views of a music collection, using an aligned SOM. This builds upon work by Logan (2002), who uses a spectrum-based similarity measure to create playlists of similar songs, and Aucouturier and Pachet (2002), who use a spectrum-based similarity measure to find similarities between songs. Lübbers (2005) describes a system called 'The Sonic SOM', which uses a SOM to help users understand their music collections. Knees (Knees, Pohle, Schedl and Widmer 2006) presents another system 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 focuses upon real-time selection of samples based on their timbral features, and is thus different from most previous timbral analysis research. Furthermore, it does not use meta-data to organise databases, allowing for greater flexibility.

As the authors are both composers, the intentions are also markedly different from MIR research: rather than attempting to navigate a search space and return an exact match, the interest is in discovering similar timbres. For example, in cases where the software is used to select timbres based on live input, the system response will 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 real-time, as the sample database must be analysed initially (see Section 3.2) and, in the case of Method 2, the SOM must be trained (see Section 5.1.1), 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 1,551 individual percussion samples; 379 Apple GarageBand loops, including pitched instrumental and unpitched percussion loops; and 83 soundscape recordings, averaging approximately 30 seconds in duration.

3.2. Sample analysis

Individual soundfiles were analysed using Jehan (2005) to derive the following time-varying feature data for each sample: brightness – the spectral centroid measure, loudness – the spectral or time-domain energy, noisiness – a bark-based spectral flatness measure, and a 24-band Bark analysis. Additionally, peak amplitude and RMS were analysed 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:

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

Lastly, the three Bark bands with the highest amplitudes, together with these values, are also stored (see Table 1). The Bark analysis is an auditory modeller that provides perceptually meaningful data, corresponding to the intensity of the first 24 critical bands of hearing. Furthermore, when compared to 512 band FFTs, the analysis itself already provides useful data reduction.

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

4. METHOD 1: FUZZY COMPARISON

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 (Klir and Yuan 1995) are used to determine closeness (see Section 4.2).

This method has been used in an existing real-time generative rhythm composition system (Eigenfeldt 2009), and is described fully elsewhere (Eigenfeldt and Pasquier 2009). It works well for short percussion samples, since it does not take into account time-varying spectra.

4.1. Organisation

In addition to the analysis file described in Section 3.2, the fuzzy comparison requires an additional

Table 1. Maximum Bark amplitudes for the three most intense bands of an example soundfile.

Band	Maximum amplitude
17	0.892
16	0.867
7	0.789

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

Table 2. An example input vector and results returned by samples_by_bands, sorted by number of direct matches between the contents and the input vector.

Input vector: Bark bands	Indices	Contents (Bark bands)	No. of matches
5 9 14	166	5 9 14	3
	76	2 9 14	2
	119	5 9 11	2
	127	9 13 14	2
	46	5 7 12	1
	73	4 9 11	1
	90	3 9 15	1
	103	9 11 14	1
	191	12 14 17	1

array to be created prior to performance: samples_by_bands is created at this time, in which pointers to spectrum_DB indices are sorted into the 24 Bark energy bands. For example, if a sample's Bark bands are (2 5 7), the spectrum_DB index is entered into samples_by_bands at all three locations. This allows access to all samples that have high energy in a specific band.

4.2. Real-time implementation

In performance, an input vector representing three Bark bands is passed to samples_by_bands, which returns an array of all sample indices whose three highest bands contain any of those values. The array's contents are then compared to the input vector, and the number of direct matches is used to sort the list (see Table 2). Those samples that are most similar, according to their highest three Bark bands, are at the top of the sorted array, while all entries will contain at least one match.

This algorithm only works if the input vector matches the analyses 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.

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). The number of direct matches between the input vector and the returned bands are summed, matches to the fuzzy list are summed and scaled by 0.66 (a hand-tuned value that created the ordering in Table 3), and the two values summed. The result is scaled to between 0 and 1 to create a closeness rating (see Table 3).

4.3. Real-time applications

One possible application for this method is the selection of samples that are similar (or dissimilar) to timbres performed live. Incoming timbres can be analysed to derive the three most intense Bark bands, and this input vector used to derive a set of similar samples. Similarly, the user can set the probabilities by hand, by selecting three bands and their relative levels (see figure 1).

Table 3. Closeness ratings based upon matching bands between results returned by samples_by_bands, and the input vector.

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

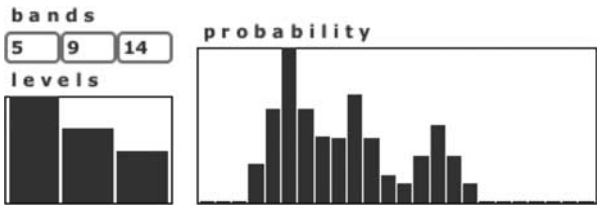


Figure 1. Defining a spectral band from which to choose a sample by setting three bands and their relative energies to determine probabilities.

A faster method used in the generative rhythm software (Eigenfeldt 2009) would be to select an approximate spectral bandwidth, from which the software can use a Gaussian probability distribution to generate a random input vector (see figure 2). This would tend to select samples whose Bark bands are close together, which is useful when discrete spectral bands are desirable.

4.4. Search heuristics

Since the search space can contain several hundred items, even when limited by using the pre-sorted samples_by_bandsarray, it was found that evaluating

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

it in its entirety (using a 16-nearest neighbour 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 1,500 ms).

Incrementing through the search space 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 generative software, which is often the case for live laptop music, this is not a desired compositional choice; indeed, different results are desirable given the same input values. Therefore, the sample list is incremented randomly until the desired number of selections are made.

When the randomly incremented search first begins, only ratings of 0.67 and higher are acceptable (see Table 3). After 250 ms, this is enlarged to include scores of 0.66 and above, therefore including samples without any direct matches, but with three fuzzy matches. After 1,000 ms, it is enlarged to include scores of 0.55 and above, including those samples with only one direct match and one fuzzy match. During the search, results are added to the samples list and loaded to be immediately available for performance.

4.5. Dissimilarity

It is often desirable to select samples that are dissimilar to a given input vector. For example, given an input timbre that contains mainly high-frequency energy, it could be useful to choose accompanying timbres that contain mainly low frequencies, as these would constitute a distinctly separate spectrum, and thus avoid timbral masking.



Figure 2. Using only a centre frequency and bandwidth to define a spectral band from which to choose a sample.

Choosing dissimilar timbres to a given Bark set is simply a matter of creating an inverse probability vector around the three bands (see figure 3a), then choosing three new bands using probabilities from this vector (see figure 3b), and finally searching for timbres with these bands.

If the dissimilar bands are generated only once prior to selection, the resulting timbres will all be related, since they share the same bands; for example, generating three dissimilar Bark bands by random selection using the probability vector from figure 3b might result in a Bark band vector of (2 12 18). If this single vector were used to select multiple samples from the database, these samples would all share that vector and thus have similar timbres. 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 high probability of dissimilarity within the new sample groups.

5. METHOD 2: SELF-ORGANISING MAP

The second method for timbral selection uses a self-organising map, or SOM (Kohonen 1998). 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 two-dimensional visualisation of the resulting organisation. In our implementation, we experimented with two types of input vectors: the three highest Bark bands (Section 5.1) and the complete 24 Bark bands (Section 5.4).

5.1. Generation of the self-organising maps

A SOM 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, scaled to between 0 and 1; however, the input vector can easily be

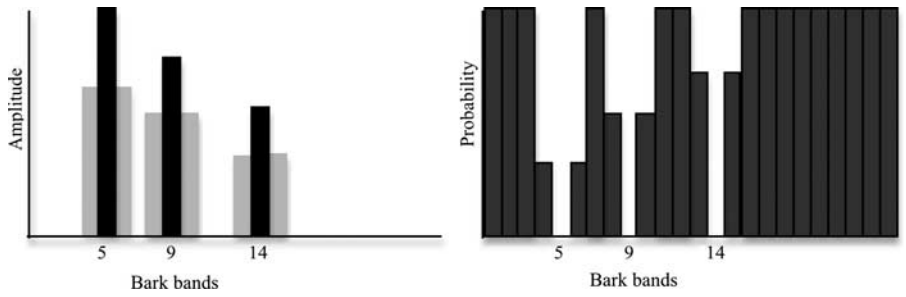


Figure 3. (a) An input vector of (5 9 14) and its 'fuzzy list' (left). (b) The inverse of this vector, used as a probability distribution for selection of dissimilar timbres (right).

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

changed to use any feature data, and be of any size (see Section 5.4). The visualisation map was a 25–by-25 Jitter matrix, representing the 625 individual nodes (artificial neurons), with the three weights of each node (one per band) mapped to RBG values. The grid itself is toroidal; that is, opposite edges are connected.

5.1.1. *Training*

Each node's weights were initialised 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 that of the map: 12 nodes in every direction. The initial learning rate (*lr*) was set to 1, and decreased at a logarithmic rate at each training iteration (*t*) using the following function:

$$lr(t) = \log_{10}(t/ti)$$

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

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

At each iteration a training example, chosen randomly from the database, is fed to the network and the Euclidean distance to all weight vectors is 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:

$$ws = lr/(ns^2 + 1)$$

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

Once training is complete, the SOM will display those samples that are similar to one another in close proximity (see figure 4). Here the shades displayed relate to the input vector (the three most intense Bark bands). Dark colours represent those samples in which all three Bark bands are low; 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 – in other words, a given Bark band vector of (17 3 5) will be bright red (not displayed in the grayscale image).

5.2. Mapping

Once training is complete, it is possible to directly associate individual nodes and input vectors. The database

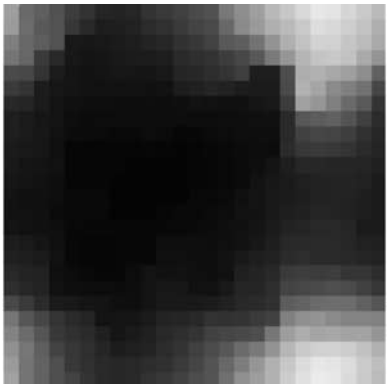


Figure 4. An example visualisation of the SOM of the percussion sample library, displayed in grayscale.

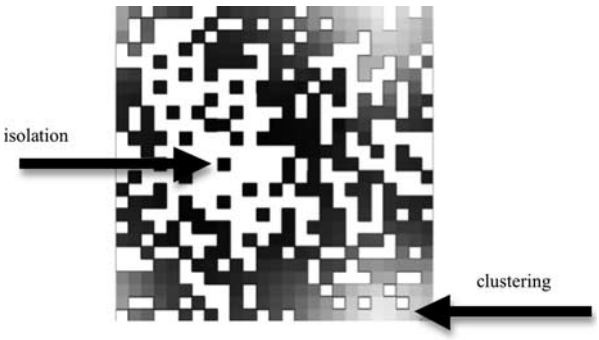


Figure 5. The associated SOM from figure 4, displaying the unassociated nodes as white. This shows which nodes are isolated, as well as those that are clustered.

was incremented in its entirety (one additional epoch), and fed to the SOM. Each BMU was used as the index into an array of matching samples.

For the SOM shown in figure 4, only 52% of the nodes had one or more samples associated with them; 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 (e.g. the three most intense Bark bands). While this demonstrates the clustering functionality of the SOM, it is impractical for the user. See Section 5.4 for a more productive use of the SOM.

The SOM can also be visualised to show only the direct associations, with non-associated nodes remaining white; thus, this visualisation shows potential clustering of data (see figure 5). Note, for example, the relative isolation of certain nodes, which signifies few similarities within the database to those samples. Compare this to the clusters of light squares at the bottom right (which also wrap around to the top of the map); this signifies a clustering of similar samples within the database. This particular visualisation does not display how many samples are associated with each

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

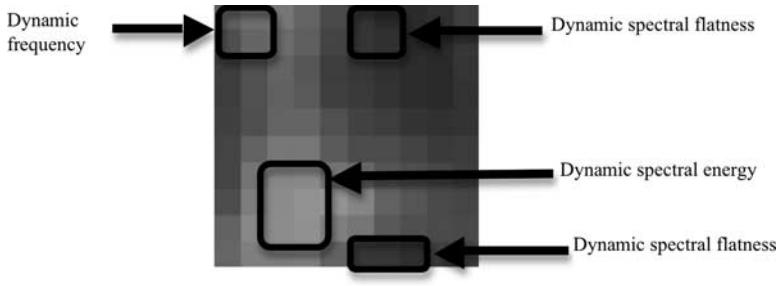


Figure 6. SOM visualising the soundscape database, using standard deviation of brightness, loudness and noise after three epochs. Since the database has fewer items, the matrix was reduced to 10×10.

node; however, implementing the z-axis for this purpose would rectify this.

5.3. Real-time implementation

An obvious benefit of using a SOM is its ability to visualise similarities within a database. The most practical real-time implementation of a SOM allows the user to click on a visual map, and derive the appropriate association(s) from the database. This is a simple matter of converting the <x,y> mouse selection coordinates in an index into the association array. Using the post-learning associative matrix (figure 4), which clearly displays learnt associations, only 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 visualisation is in the choice of features displayed. It was found that the best features to choose were those that had significant differences within the database.

For example, figure 6 displays the soundscape database, using the standard deviation of the sample's brightness, loudness, and noise over its duration. Squares that display red are associated with samples that have greater dynamic frequency (spectral centroid) change; squares that display green have greater variation in their spectral energy; those squares that display blue have greater dynamic change in their spectral flatness. These areas in the grayscale SOM are highlighted (see figure 6).

Figure 7 displays the selected GarageBand loop database using mean brightness, standard deviation of brightness and mean frequency. In this case, dark squares contain low frequencies, and have little change in their spectral centroid (i.e. the bass samples), the brightest squares have higher spectral energy, with dynamic spectral change (i.e. percussion samples).

5.5. Larger feature sets

Since the Bark analysis produces 24 discrete intensity values, it was possible to train the SOM using all

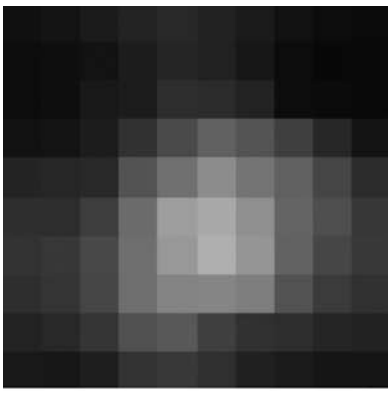


Figure 7. SOM visualising the GarageBand loop database using mean and standard deviation for brightness, and mean frequency.

24 values as an input vector. Using the GarageBand loops, after 64 epochs (about 100,000 training iterations), finer delineations between data 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.

Visualising 24 inputs in two dimensions required mapping the 24 weights of each node to the three RGB values. The averages over eight bands' intervals were used. As a result, red squares indicate energy primarily in low-frequency bands, green squares indicate energy in the mid-range frequency bands, and blue squares indicate energy in the high-frequency bands (see a simplified grayscale version in figure 8).

5.6. Real-time applications

It was found that the most effective use of SOMs in performance displayed several visualisations at once, allowing the user to make selections in one or more of the maps and correlate the resulting selections. For example, by using one SOM to display temporal information (such as figure 5), and one to display frequency content, it is possible to select items from the database that are associated with both selections.

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

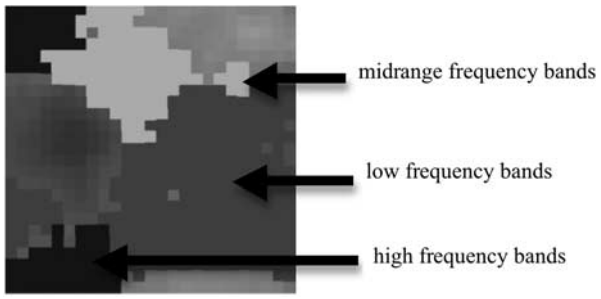


Figure 8. SOM visualising 24 Bark bands of GarageBand loops using mean amplitude.

Another useful application was to draw paths through the visualised data, to be interpreted over time. For example, using the visualisation in figure 5, 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 dynamic noisiness.

6. COMPARISON

Both methods were found to be useful, each offering benefits and detriments that need to be considered depending on a specific performance's 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 are required, this is of great benefit; for example, in the case of Eigenfeldt (2009), which utilises a set number of agent-based performers. The method offers a very simple and efficient selection mechanism. The user can choose a type of timbre by specifying dominant spectral energy, either through three bands (as in figure 1), or a frequency centre and bandwidth (as in figure 2), and move towards a similar or dissimilar sound in one click.

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 visualise the database, it is possible to define criteria that do not exist within the database. This is especially true if the database is relatively small. While our aim is to investigate real-time sample selection based mainly on timbral similarity, this method reduces it to very basic spectral information. As is often the case, there is a trade-off between user interface simplicity and quality of the control obtained.

6.3. Benefits to Method 2

SOMs visualise 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 visualised 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 real-time performers is no actual calculations – other than conversion of mouse location to database indices – are necessary: all intensive calculations are done prior to performance.

6.4. Detriments to Method 2

This is a very generic method, as opposed to the ad hoc heuristic approach of Method 1. Although SOMs allow easy visualisation, it is not immediately clear what is being visualised; for example, knowing what a red pixel represents, as opposed to a blue, requires an understanding of the analysis itself. Of course, this can be easily addressed by providing the user with the proper documentation, and designing a user interface that balances simplicity and control.

Since SOMs involve time-consuming training, they are not real-time, and therefore 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 real-time sample selection based upon timbral properties and similarities within large databases of samples. Both methods offer different benefits to the laptop performer and both methods can be implemented within MaxMSP.

Several future directions are currently being 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 avoided by borrowing node locations from similarly rated samples: for example, a sample with a Bark band vector of (5 9 14) could be inserted into nodes that contain the identical vector. 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, as described in Pampalk (2004). Furthermore, balanced clustering could be applied in which

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press

all nodes have the same number of samples, or the size of the map could be automatically adjusted to the size of the library. Multi-layered (Rahman, Pi, Chow and Wu 2007) and hierarchical SOMS (Luttrell 1989; Lampinen and Oja 1992) could also be implemented.

The various software systems introduced in this paper, along with referenced generative rhythm software, were created in Max/MSP, and are available at the first author's website: www.sfu.ca/~eigenfel/research.html.

REFERENCES

Aucouturier, J.-J. and Pachet, F. 2002. Music Similarity Measures: What's the Use? *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. Paris: IRCAM, 157–63.

Cano, P. and Koppenberger, M. 2004. Automatic Sound Annotation. *IEEE Workshop on Machine Learning for Signal Processing*. Sao Luís: IEEE, 391–400.

Ciufo, T. 2005. Beginner's Mind: An Environment for Sonic Improvisation. *Proceedings of the International Computer Music Conference (ICMC)*. Barcelona/San Francisco: ICMA, 781–4.

Eigenfeldt, A. 2009. The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine. *Applications of Evolutionary Computing*. Berlin: Springer, 498–507.

Eigenfeldt, A. and Pasquier, P. 2009. Realtime Selection of Percussion Samples Through Timbral Similarity in Max/MSP. *Proceedings of the International Computer Music Conference (ICMC)*. Montreal/San Francisco: ICMA, 77–80.

Fiebrink, R., Wang, G. and Cook, P. 2008. Support for MIR Prototyping and Realtime applications in the ChucK Programming Language. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. Philadelphia, 153–60.

Hsu, W. 2006. Managing Gesture and Timbre for Analysis and Instrument Control in an Interactive Environment. *Proceedings of the International Conference on New Interfaces for Musical Expression*. Paris: IRCAM, 376–9.

Jehan, T. 2005. *Creating Music by Listening*. PhD thesis. MIT Media Lab, Cambridge, MA. http://web.media.mit.edu/~tristan/phd (accessed on 8 December 2009).

Jehan, T. and Schoner, B. 2001. An Audio-Driven Perceptually Meaningful Timbre Synthesizer. *Proceedings of the International Computer Music Conference (ICMC)*. Havana/San Francisco: ICMA, 381–8.

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

Knees, P., Pohle, T., Schedl, M. and Widmer, G. 2006. Automatically Describing Music on a Map. *Workshop on Learning the Semantics of Audio Signals*. Athens, 33–42.

Kohonen, T. 1998. The Self-Organizing Map. *Neurocomputing* **1–3**(6): 1–6.

Lampinen, J. and Oja, E. 1992. Clustering Properties of Hierarchical Self-Organizing Maps. *Journal of Mathematical Imaging and Vision* **2**(2–3): 261–72.

Lippe, C. 1993. A Composition for Clarinet and Realtime Signal Processing: Using Max on the IRCAM Signal Processing Workstation. *Proceedings of the 10th Italian Colloquium on Computer Music*. Milan, 428–32.

Logan, B. 2002. Content-Based Playlist Generation: Exploratory Experiments. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. Paris: IRCAM, 295–6.

Lübbers, D. 2005. Sonixplorer: Combining Visualization and Auralization for Content-Based Exploration of Music Collections. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. London, 590–3.

Luttrell, S. P. 1989. Hierarchical Self-Organizing Networks. *Proceedings of the 1st IEE Conference on Artificial Neural Networks*. London, 405–13.

Pampalk, E., Dixon, S. and Widmer, G. 2004. Exploring Music Collections by Browsing Different Views. *Computer Music Journal* **28**(2): 49–62.

Rahman, M. K., Pi Yang, W., Chow, T. W. and Wu, S. 2007. A Flexible Multi-Layer Self-Organizing Map for Generic Processing of Tree-Structured Data. *Pattern Recognition* **40**(5): 1406–24.

Rebelo, P. and Renaud, A. 2006. The Frequencyliator: Distributing Structures for Networked Laptop Improvisation. *Proceedings of the International Conference on New Interfaces for Musical Expression*. Paris: IRCAM, 53–6.

Tzanetakis, G. and Cook, P. 2000. MARSYAS: A Framework for Audio Analysis. *Organised Sound* **4**(3): 169–75.

Zwicker, E. 1961. Subdivision of the Audible Frequency Range into Critical Bands. *Journal of the Acoustic Society of America* **33**(2): 248.

https://doi.org/10.1017/S1355771810000154 Published online by Cambridge University Press