MASOM: A Musical Agent Architecture based on Self-Organizing Maps, Affective Computing, and Variable Markov Models

Kıvanç TATAR and **Philippe PASQUIER**
School of Interactive Arts + Technology, Simon Fraser University
Surrey, BC, CANADA
{ktatar, pasquier}@sfu.ca

Abstract

Musical Agent based on Self-Organizing Maps (MASOM) is a machine improvisation software for live performance. MASOM plays experimental music and free improvisation. The agent perceives and generates audio signals. MASOM combines Self-Organizing Maps for sound memory, Variable Markov Models for musical structure, and Affective Computing for machine listening. The agent learns the sonic content and the musical structure to generate live performances. MASOM's offline learning uses an audio corpus of recordings of performances or compositions. The machine listening module of MASOM extracts high-level features such as eventfulness, pleasantness, and timbre. The agent listens to itself and other performers to decide what to play next.

Introduction

Metacreation is the idea of endowing machines with creative behaviors (Pasquier et al. 2017). Metacreation applies the knowledge of Artificial Intelligence to develop autonomous systems solving creative tasks. *Musical Metacreation* (MUME) is a sub-branch of Metacreation. MUME focuses on the creative tasks of music. Autonomy and agency being essential components of Metacreation and MUME, artificial agents become the perfect modeling paradigm.

In this study, we present a new musical agent architecture. An agent is a proactive system that autonomously initiates actions to respond to its environment in timely fashion (Wooldridge 2009). Agents work both online and offline. While a variety of musical tasks have been addressed by Multi-agent Systems (MAS), we focus on improvisation in experimental electronic music in this study.

A *musical agent* is an autonomous system that creates music or a part of the music, individually or in a community of agents. MASOM is a flexible musical agent that only requires a corpus of recordings for the learning and an audio signal as an input to listen to other agents. MASOM implements a hybrid agent architecture that combines Self-Organizing Maps as a musical memory, Variable Order Markov Models for pattern recognition and generation in music, and Affective Computing and machine listening to model human hearing.

MASOM's architecture stands out with the following contributions in musical agents:

- The use of SOMs as a musical memory of audio samples
- The capacity of a musical agent that can listen to big data of music
- The introduction of sound affect estimation in offline and online machine listening
- The flexibility of a musical agent to perform alone or with other agents, software or human.
- Machine listening with the time scales of *micro*, *sound object*, and *meso*

Roads (2004) proposes *infinitesimal, subsample, sample, micro, sound object, meso, macro, supra,* and *infinite* time scales of music, arranged from the one with the shortest duration to the longest one. MASOM inherits three different time scales of *micro, sound object*, and *meso*. *Micro* time scale spans from a millisecond to approximately 100ms. The duration of sound objects varies from a fraction of a second to several seconds. *Meso* time scale ranges from seconds to minutes. We explain how these time scales are incorporated in MASOM's architecture in the Section System Design.

Background

Self-Organizing Maps

Self-Organizing Maps (SOMs) are artificial neural network models, inspired by neurophysiology (Kohonen 1998). SOMs visualize, represent, and cluster high-dimensional input data with a simpler 2D topology. SOM topologies are typically square and include a finite number of nodes. Node vectors have the same number of dimensions as the input data. SOMs organize the input data using a 2D similarity grid so that similar data clusters locate closer to each other in the topology. Moreover, SOMs cluster the input data by assigning each input vector to the closest node called the best matching unit (BMU). Figure 1 shows a SOM with 100 nodes. Each square represents one node that is colored according to its feature value.

The training is unsupervised in SOMs, but designers set the topology and the number of nodes in the topology. Each input vector is a training instance of SOM's learning.

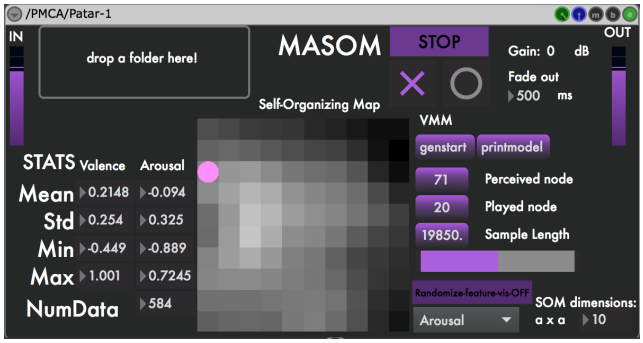

Figure 1: The generation interface of MASOM includes a visualization of SOM. The visualization shows one dimension at a time. The dimensions are normalized between -1.0 (black) and 1.0 (white) for the visualization.

There are three ways to initialize SOM nodes: starting with zero vectors, randomizing model vectors, and using principal component analysis on the input vectors. During a training instance, a SOM also updates BMU's neighboring model vectors using a *neighborhood function*. Common *neighborhood functions* are Gaussian, cut-Gaussian curves, linear, and piecewise linear functions.

On each training instance, SOMs update their nodes using the data of an input vector. First, SOMs find the BMU of an input vector. Second, SOMs calculate the Euclidean distance between the input vector and the BMU. Third, SOMs update the BMU by this distance multiplied by the *learning rate*. The *learning rate* is a user-set global parameter in the range $[0., 1.]$. Lower learning rate corresponds to less adaptive and more history-depended SOMs. Depending on the neighboring function, SOM also updates the neighbors of BMU in the direction of BMU's update. The update amount becomes less as the neighboring node is further away from the BMU. Therefore, the BMU and its neighboring nodes move closer to input vectors on each training instance.

It is common to use the same dataset more than once to train a SOM. Each pass on the dataset is called an *epoch*. The learning rate is adaptive, decreasing as the number of epoch increases. Hence, model vectors change less as the number of epochs increases (Kohonen 1998). The training of SOM stops after some epochs, or when the nodes are updated less than a user-set amount, or not updated at all.

Markov Models

Markov Models are finite state machines that model patterns in discrete sequences through the Markov assumption. An N^{th} order Markov model assumes that,

$$P(s_t|s_{t-1}, s_{t-2}, ..., s_1) = P(s_t|s_{t-1}, ..., s_{max(t-N,1)}) \quad (1)$$

Hence, Markov models are history dependent. A first order Markov model depends on the current state to predict the next. A second order Markov model takes the current and the previous state into account to predict the next state. That is, the order of a Markov model points out how many previous states to be considered to predict and generate the next state. Moreover, the observed number of transitions between Markov states determines the conditional probabilities of the transitions. Therefore, a Markov Model is a stochastic model represented as a directional graph.

Variable Markov models (VMMs) refer to a family of algorithms including probabilistic finite automata, probabilistic suffix automata, prediction suffix trees, Lempel-Ziv 78, improved Lempel-Ziv (LZ-MS), prediction by partial match (PPM), Factor Oracles, and the context tree weighting method (Ron, Singer, and Tishby 1996; Begleiter, El-Yaniv, and Yona 2004). VMM considers a varying number of previous states to predict the next state. Markov Models are applied to a variety of MUME tasks including the musical agent design (see Section Musical Agents with Markov Models).

Affective Computing

Eerola and Vuoskoski (2013) mention that "...the emotional effects of music are the most important reason why people engage in musical activities." and categorize affect models in the literature in four classes: *discrete*, *dimensional*, *miscellaneous* and *music-specific*. First, the discrete affect models assume that one can explain all emotions using a finite set of basic emotions such as happiness, sadness, fear, anger, and disgust (Ekman 1992; Panksepp 1998) as well as shame, embarrassment, contempt and guilt (Ortony and Turner 1990). Second, the dimensional affect models represent emotions using two or more continuous dimensions that are ideally orthogonal to each other. The most common dimensional affect model has two dimensions: valence (pleasantness) and arousal (eventfulness). Some dimensional affect models include additional dimensions such as tension, potency, dominance (Eerola and Vuoskoski 2011). The continuous circumplex model is an example of a dimensional affect model (Russell 1980). Music Information Retrieval (MIR) studies frequently use the dimensional affect model (Eerola and Vuoskoski 2013). Third, miscellaneous models are collections of concepts that are linked to emotions such as intensity, preference, similarity, tension. Fourth, music-specific affect models are proposed as emotion lists that are specifically relevant to music (Zentner, Grandjean, and Scherer 2008). The discussions around a list of music-specific emotions are still ongoing (Eerola and Vuoskoski 2013).

The cognition of affect has different layers. Livingstone et al. (2007) model three different layers in the cognition of emotion in sound and music: perceived, induced and expressed emotion. The perceived emotion is the subjective perception of stimuli. The perceived emotion goes through a cognitive process and becomes the reaction that is the induced emotion. The expressed emotion is the conveyed emotion that is stimulated by subjects (humans). In this study, the affect estimation in sound focuses on the perceived emotion. We explain our multivariate regression algorithm for affective computing in sound in the Section Sound Affect Estimation.

Related Work

In this section, we explain why we choose SOMs to model musical memory and mention a musical agent that use a SOM in the system design. Then, we briefly cover musical agents with Variable Markov Models.

Modelling Musical Memory with SOMs

Gabora (2002) proposes three properties of the memory in the cognitive processes of creativity:

- Memory is sparse
- Memory is distributed, but distributions are constrained
- Memory is content addressable

Bogart and Pasquier (2013) build on Gabora's work to model the memory of creative visual processes using SOMs. Bogart and Pasquier propose that SOMs are beneficial to model the connection between the sensory input and the field experience of an agent. SOMs satisfy the three memory properties of creative processes proposed by Gabora. If we model the creative memory with SOM, the memory is sparse since SOMs consist of separate node vectors arranged in a 2D plane. The memory is distributed but distributions are constrained because SOMs have a finite number of nodes that are constrained by the domain represented by the sensory input. Memory is also content addressable in SOMs. SOM nodes represent the clusters the sensory input instances. Moreover, the node vectors are not the exact replications of sensory input vectors although SOM nodes are aligned by the sensory input.

Regarding the applications of SOMs as the memory of a musical agent, we have found only one implementation in the literature. Smith and Deal (2014) use SOMs in the short term memory of a musical agent. This musical agent works with audio inputs and extracts audio features of chroma, brightness, noisiness, and loudness. There are two layers in the memory: long-term memory and adaptive memory. The long-term memory is a k-d tree trained on audio feature vectors at the end of each performance sessions. Also, each input audio feature vector is a search query of the k-d tree during the performance. The agent trains and updates the SOM online using chroma vectors of the input. The amount of change in SOM node vectors in a control signal that is passed to the *decision* module. The decision module uses this distance to deviate from the input feature vector to introduce variance in the agent's output.

SOMs have also been used to organize large collections of audio samples (Eigenfeldt and Pasquier 2010; Fried, Jin, and Oda 2014). Hence, we decided to focus on SOMs as the sound object memory of MASOM.

Musical Agents with Markov Models

Markov Models have been extensively used in musical agent design because of their success on the prediction (Begleiter, El-Yaniv, and Yona 2004) and generation (Pachet 2003) of symbolic music sequences.

The *Continuator* is a musical agent working with a symbolic representation of music (Pachet 2003). The Continuator uses VMM to continue a musical phrase. Another musical agent with VMM is *Beatback* (Hawryshkewich, Pasquier, and Eigenfeldt 2010). Beatback uses VMM to generate rhythms. Moreover, Factor Oracle, an algorithm that is similar to VMM, has been extensively applied to musical agents design, including *OMAX, Audio Oracle, PyOracle*, *Improtek*, and *Variable Markov Oracles* (VMO) systems (Assayag and Dubnov 2004; Assayag et al. 2006; Dubnov, Assayag, and Cont 2007; Lévy, Bloch, and Assayag 2012; Nika and Chemillier 2012; Surges and Dubnov 2013; Wang and Dubnov 2014; Nika et al. 2015; Wang, Hsu, and Dubnov 2017). Amongst all these musical agents, VMO is the closest architecture to MASOM. Similar to MASOM, VMO can perform live with or without other human or software agents. VMO differs from previous FO implementations. In the previous FO implementations, each state represents a musical phrase segment that is symbolic. However, each state in VMO is a cluster of audio frames. VMO uses a distance *threshold* to cluster audio frames. If the distance between feature vectors of two audio frames is less than the threshold, these frames are added to the same cluster. The agent sets the distance threshold automatically by calculating the threshold value that gives the highest *information rate* (IR). IR is extensively used in Pattern Matching and Recognition studies as a measure of information content.

System Design

This section begins with the explanation of the affect estimation algorithm and machine listening in MASOM. We continue by presenting the learning and generation in MASOM.

Sound Affect Estimation

Affect estimation in sound and music is still an open problem (Eerola and Vuoskoski 2013). Fan, Thorogood, and Pasquier (2016) present a machine learning model that estimates pleasantness and eventfulness of soundspace recordings. The authors implement a 2D continuous affect model proposed by Russell (1980). Using multivariate linear regression, their model is trained on a data set of 125 soundscape samples with 6-second duration. The data set is labeled by an online study with 20 participants. In this study, we use the same dataset that the study of Fan, Thorogood, and Pasquier (2016) uses. We used a different audio feature extraction library and we applied multivariate regression to generate an affect estimation model. We implement feature extraction in MAX[1] using *ircamdescriptor~* object provided in MAX Sound Box externals[2]. MAX provides opportunities to use the affect estimation system for both offline affect estimation of an audio corpus and realtime affect estimation in machine listening applications. Next, we explain each audio feature used in the affect estimation model and introduce the model.

All audio features are computed with a window size of 1024 samples (23ms) and a hop size of 512 samples (12ms),

[1] https://cycling74.com/

[2] http://forumnet.ircam.fr/shop/en/forumnet/53-max-sound-box.html.

which correspond to the *micro* time scale. We calculate the mean and standard deviation of these audio features over a moving window of 6-seconds (Fan, Thorogood, and Pasquier 2016) which corresponds to the sound object time scale. There are five audio features included in the affect estimation model:

- *Mel Frequency Cepstral Coefficient* (MFCC) is a known feature in audio processing (Peeters 2004). MFFC calculation combines Mel frequency scale with a particular frequency spectrum calculation called *cepstrum*. Mel frequency scale represents the critical bands of human hearing. The cepstrum stands for the discrete cosine transform (DCT) of the logarithm of the spectrum (FFT). There are 13 MFCCs in our calculation excluding the zero coefficient. $MFCC0$, the energy, or the DC offset is removed.
- The second feature that we use in the affect estimation is *loudness*. We use the algorithm proposed by Moore, Glasberg, and Baer (1997) to calculate the loudness.
- *Spectral Flatness* is the ratio of geometric mean to the arithmetic mean of the energy spectrum. Spectral flatness shows the noisiness against sinusoidality of the spectrum. We compute the spectral flatness in four bands: 250 - 500, 500 - 1000, 1000 - 2000, 2000 - 4000 Hz. $SpectralFlatness1Mean$ in equation 2a refers to the moving average of the spectral flatness computer over the band 250 - 500 Hz.
- *Perceptual Spectral Decrease* is the amount of decreasing of the spectral amplitude, computed using a human hearing model (Peeters 2004).
- *Tristimulus* is the calculation of three different types of energy ratio (Peeters 2004). *Perceptual Tristimulus* uses a human hearing model to calculate tristimulus.

Equation 2a and 2b introduce our affect estimation model generated by the multivariate linear regression:

$$\begin{aligned} Valence = &-0.169 + \\ &-0.061 * LoudnessMean \\ &+0.588 * SpectralFlatness1Mean \\ &+0.302 * MFCC1STD \\ &+0.361 * MFCC5STD \\ &-0.229 * PerceptualSpectralDecreaseSTD \end{aligned} \tag{2a}$$

$$\begin{aligned} Arousal = &-1.551 \\ &+0.060 * LoudnessMean \\ &+0.087 * LoudnessSTD \\ &+1.905 * PerceptualTristimulus2STD \\ &+0.698 * PerceptualTristimulus3Mean \\ &+0.560 * MFCC3STD \\ &-0.421 * MFCC5STD \\ &+1.164 * MFCC11STD \end{aligned} \tag{2b}$$

We use the affect estimation model given in Equation 2a and 2b within two sub-modules: offline labeling of automatic corpus generation in the learning module and online machine listening in the generation module. For example, Figure 2 exemplifies the affective labels of an audio corpus. We labeled each audio segment using the equation 2a and 2b. We explain the details of the audio segmentation in the following section.

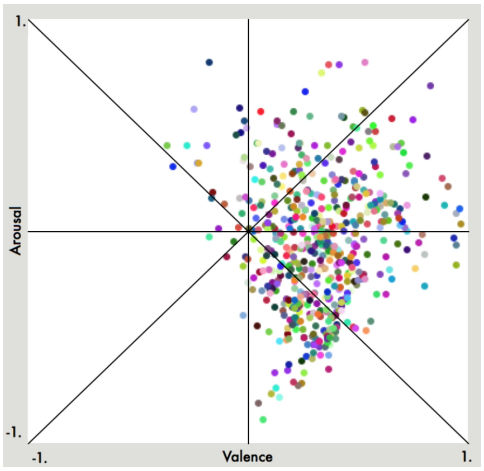

Figure 2: Each dot represents a novel segment of Stockhausen's Kontakte. Each segment is labeled using the dimensional affect estimation model of MASOM.

The Learning in MASOM

Automatic Corpus Generation MASOM automatically generates its memory using an audio corpus. There are two steps of creating the audio samples in the memory: *segmentation* and *labeling* (Figure 3).

We segment an audio file using the MIRToolbox[3] library in MATLAB. MASOM applies a new segmentation algorithm based on novelty, called *Multi-granular approach* (Lartillot et al. 2013). The MIRToolbox includes this new segmentation algorithm. This approach calculates a novelty curve using the similarity matrix that displays the musical structures in an audio file. The generation of the similarity matrix has two steps. First, the algorithm generates a dissimilarity matrix by calculating the distances between an audio frame and all previous frames. The choice of the type of distance measure relies on the audio feature in focus. Cosine and Euclidean are common distance measures in the similarity matrix calculations. Second, the algorithm converts a dissimilarity to a similarity matrix using one of two equations: linear, $y = 1 - x$ or exponential, $y = exp(-x)$. To generate the similarity matrix, MASOM uses Fast Fourier Transform (FFT) with 50ms windows with no overlapping to calculate the spectrum. Then, MASOM calculates cosine distances between a frame and its preceding frames to generate the dissimilarity matrix. Lastly, the agent uses the linear conversion to convert the dissimilarity matrix to the similarity matrix.

Using the similarity matrix, the segmentation algorithm calculates of a novelty curve. A novelty curve is the probability of transitions between successive sound objects. The local maxima in novelty curves indicate a high probability of transitions, and therefore, segmentation points. The segmentation procedure ends by saving each segment as a different file to generate an audio corpus of

[3] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

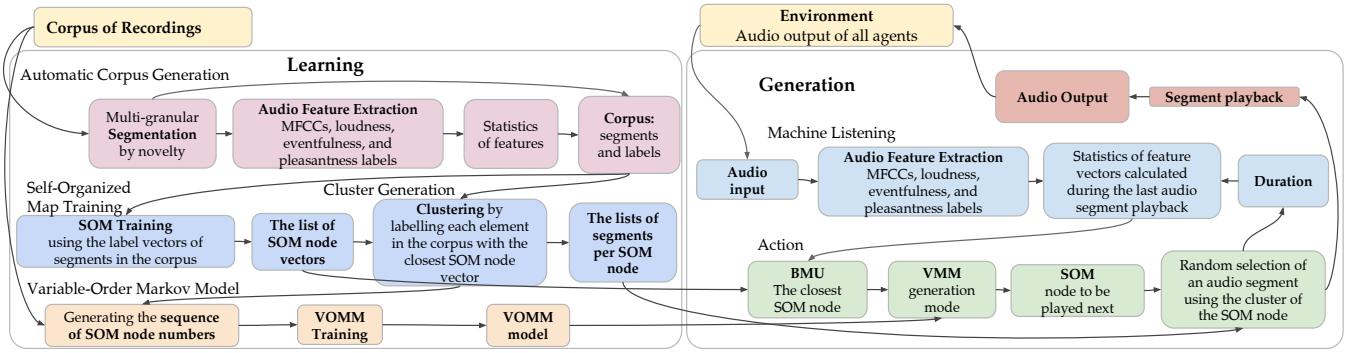

Figure 3: The architecture of MASOM

MASOM's musical memory. Figure 4 shows an example of MASOM's segmentation. In our experiments with a variety of corpus, we observe that this segmentation algorithm successfully creates audio segments ranging from a fraction of a second to several seconds, which corresponds to the sound object time scale.

Following, we label each audio segment with a vector with 31 dimensions. The average and standard deviation of the audio features are computed over the whole segment. Sixteen dimensions are the average of 2 affect estimation features, 13 MFCCs, and loudness. Fourteen dimensions are the standard deviations of 13 MFCCs and loudness. The remaining dimension is the length of the audio segment in seconds. These labels are later used as feature vectors to train the SOM.

The Self-Organizing Map We use the *ml.som* MAX object to implement the SOM. The object is publicly available by Smith and Garnett (2012). Input vectors of the SOM are 31-dimensional vectors of audio segments in the corpus. The topology of the SOM is rectangular. The size of the topology changes with the number of audio files in the corpus. The SOM topology is $a * a$, where

$$a = int(\sqrt{\text{the number of audio samples in the memory}/6}) \quad (3)$$

Hence, the total number of SOM nodes is approximately one-sixth of the total number of audio files. The total number of epochs is 400 in the training of the SOM. The learning rate is initially set to 0.25 and linearly decreases to 0.001 as the epoch step increases. The neighborhood function is linear, that is, the amount of update is linearly decreasing as the neighboring node is further away from the BMU. The neighborhood radius is initially set to $r = a/2$ and linearly decreases to $r = 1$ as the epoch step increases. We came up with the Equation 3 and the parameters of SOMs after several trials with corpora.

As a result of the SOM training, MASOM uses the SOM to generate clusters in the corpus. MASOM labels each audio segment with its BMU. Hence, each SOM node represents a cluster of audio segments. Some nodes may end up with no audio segments after clustering. In our trials, we found the ratio of 6 in equation 3 by aiming for the least number of SOM nodes without any audio segments. We further discuss this in the Section Evaluation and Future Work and mention our future work to develop MASOM's memory further.

The Variable-Order Markov Model MASOM trains a VMM using a string of SOM nodes. First, MASOM labels each audio segment with its BMU in the trained SOM. Second, MASOM uses the original order of the audio segments to create a string of SOM nodes. Then, MASOM trains the VMM using this string.

For example, Figure 4 shows the waveform of the seventh track of Bernard Parmegiani's *De Natura Sonorum* album. The track is a minute and forty-three seconds long. Although this track is too short to train MASOM, we exemplify MASOM's training using such a short track. First, MASOM creates novel segments of the track. Each block in Figure 4 represents one segment. MASOM found 32 novel segments in this track. Second, MASOM labels each audio segment with a 31-dimensional vector. Third, MASOM sets the topology of the SOM as 3x3 using the equation 3. Fourth, MASOM trains the SOM using the label vectors of audio segments. Fifth, MASOM labels each audio segment with its BMU. The BMUs of audio segments in this track is written on each audio segment in Figure 4. Using the original order of the audio segments, we create a string of SOM nodes. For this track, the SOM node string is,

$$885774001626368852321120414117 73 \quad (4)$$

Lastly, MASOM uses this string to train VMM.

We implement VMM in MAX using the VMM java external for MAX [4]. This external is an implementation of Prediction by Partial Match-Method C (PPM-C) algorithm. PPM-C requires a pre-set maximum Markov order. In MASOM's architecture, the maximum Markov order is 10. This is because we designed VMM to work in meso time scale of music. Our segmentation algorithm that we explain in the Section Automatic Corpus Generation generates segments ranging from a fraction of a second to several seconds, which is sound object time scale. Hence, a maximum of 10 segments would range from seconds to minutes, which corresponds to the *meso* time scale.

[4]VMM java external for MAX is available at `http://www.am-process.org/main/?portfolio=vmm`

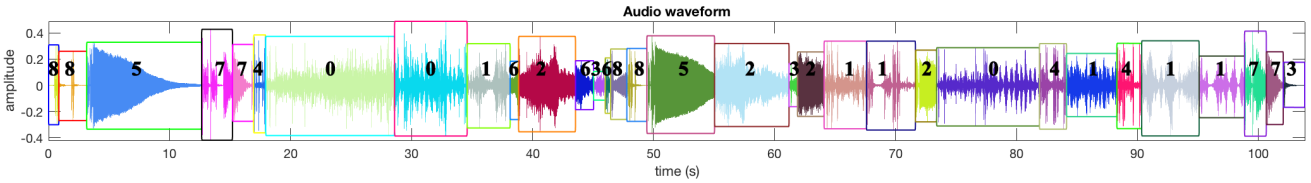

Figure 4: The waveform is the seventh track of Bernard Parmegiani's *De Natura Sonorum* album. Each square indicates a novel segment. The corresponding SOM node of the segment is written inside the box.

Following, we explain the details of the PPM-C algorithm. The performance of a compression algorithm is tested by calculating the log-loss over a test sequence. Lower log-loss over a test sequence implies better compression rates (Begleiter, El-Yaniv, and Yona 2004). However, the probability of an unobserved sequence is zero. Hence, the log-loss of such sequence is infinite. This is known as the zero frequency problem in pattern matching and recognition. PPM-C handles zero frequency using the escape method.

The escape method is as follows. Given a training sequence with length D, for each context s with length $k \leq D$, PPM partitions a total probability, $P_k(escape|s)$ between all symbols that does not appear after the context s. PPM allocates the remaining probability, $1 - P_k(escape|s)$ between the symbols that appear after the context s. The PPM variant is determined by how $P_k(escape|s)$ is calculated and how $1 - P_k(escape|s)$ is distributed amongst the symbols with non-zero counts (Begleiter, El-Yaniv, and Yona 2004).

In particular, the escape mechanism of PPM-C is as follows. Given a maximum VMM order n, context s, and symbol σ,

$$P(\sigma|s) = \frac{f_\sigma}{L+M} \tag{5a}$$

$$P(escape|s) = \frac{M}{L+M} \tag{5b}$$

where M is the number of unique symbols in the alphabet, L is the sum of frequency counts of all symbols in s, and f_σ is the frequency count of a symbol σ . When the escape mechanism happens, PPM-C decreases the order by 1 and calculates the probabilities for the order $n-1$.

We choose the PPM-C in MASOM's implementation because PPM-C and Decomposed Context Tree Weighting (DE-CTW) are shown to outperform other compression algorithms that are Binary-CTW, Lempel-Ziv 78 (LZ78), improved LZ78 (LZ-MS), and Probabilistic Suffix Trees on predicting MIDI sequences of well-known classical and jazz pieces (Begleiter, El-Yaniv, and Yona 2004). We further discuss this in the Section Evaluation and Future Work.

The Generation in MASOM

MASOM's generation module includes two submodules: online machine listening and musical action, as depicted in Figure 3. The environment of the agent is the summation of the audio output of all agents, software or human. MASOM perceives the current musical state of the environment using its musical memory and the online machine listening.

The machine listening module implements feature extraction, affect estimation, and calculation of statistics. The affect estimation algorithm in the generation module is the online version of the algorithm that we explain in the Section Sound Affect Estimation. MASOM calculates the statistics of audio features within the duration of the sample played by the musical action module. When the action module triggers a new sample, the machine listening module clears all statistics. The machine listening module outputs a 31-dimensional vector to the action module.

The musical action module calculates the distances between the vector provided by the machine listening module and the SOM nodes of the agent to find the BMU. This BMU represents the current musical state that is perceived by MASOM. Then, the action module sends the BMU to VMM. VMM keeps track of the history the BMUs to create a context s. Using this context, VMM predicts a SOM node to be played next. We clarify in the Section The Self-Organizing Map that each SOM node represents a cluster of audio segments. When the previous sample playback finishes, MASOM uses the SOM node predicted by VMM to decide on a cluster of audio segments. Lastly, MASOM randomly chooses a sample within the cluster node to generate the audio output.

Evaluation and Future Work

Examples of MASOM's output are available online[5]. The online content includes a MASOM trained on Parmegiani's De Natura Sonorum album. We also provide a recording in which audio segments of this album are played randomly. For now, we let our readers decide the success of MASOM by comparing the random playback of audio segments to MASOM's output in self-listening mode. The online content also includes documentation of MASOM's public performances with human performers and other MASOM agents. Moreover, the first step of our future work is running evaluation experiments.

As of May 2017, MASOM has performed in various venues in Vancouver, Canada; and Ístanbul, Turkey. The early performances of MASOM were free improvisation in noise music, in a duo setting with the first author. For the first public concert in October 2016, MASOM was trained on a noise album of the first author. The first author commented that MASOM was successful at copying the musical style

[5] http://metacreation.net/masom/

of the album. The first author also found that playing in a duo setting with the agent was more satisfying than playing solo and the agent was successful at proposing new musical ideas when the human performer ran out of improvisational ideas. Moreover, the first author emphasized that a stereo performance setting in which the agent was on one channel whereas the human performer was on the other, improved the clarification of the communication between the agent and the human performer. After this public concert, some audience members commented that they would like to see a visualization of the agent. In our future work, we plan to visualize MASOM.

For the second concert in December 2016, we pushed MASOM's capabilities with a concert with the NOW Ensemble. The ensemble includes saxophone, trumpet, piano, double bass, and drums. NOW Ensemble plays experimental music, free improvisation, and structured improvisation. MASOM was trained of a recording of NOW Ensemble for this second concert. This performance was a challenge because the agent listening was through a microphone instead of a line signal. The acoustic instruments were not amplified and the distance of the instruments affected what the agent was listening. Such setting requires the mixing of acoustic instruments so that the agent listens to a balanced mix of all instruments. In our future work, we plan to study automatic mixing for the machine listening of musical agents.

The third performance of MASOM was again free improvisation in noise music. This performance was in İstanbul in December 2016. The performance had three sections. The first section was a duo of the first author and MASOM. The second section was a duo of two different MASOM agents. The third section was a trio of two MASOM agents and the first author. Some of the audiences informally reported that they preferred the second section without the human performer to the other sections. This three-section piece was also performed in Vancouver, Canada in March 2017.

The fifth performance of MASOM was again in Vancouver in April 2017. The context of this performance was structured improvisation in electro-acoustic music, including MASOM, the first author, and the second author as the performers. The performance had three different sections in which different MASOM agents were playing. The agents were separately trained on Bernard Parmegiani, David Tudor, and Ryoji Ikeda. The studio session of the rehearsal of this performance is available online. Comparing two different takes with Ryoji Ikeda corpus, we recognize that the agent played louder and noisier when the human performers played louder and noisier overall.

The machine learning model that we use for affect estimation in sound is trained on a corpus of soundscape recordings (Fan, Thorogood, and Pasquier 2016). Although our experiments with this affect estimation model are convincing, we want to develop a new machine learning model using a corpus of experimental electronic music excerpts. We are in the process of designing an empirical evaluation experiment in which participants rank experimental electronic music excerpts on a continuous 2D affective grid. Using such data, we aim to develop a new model to estimate affect of sounds used in experimental electronic music.

In this version of MASOM, the SOM is MASOM's symbolic memory. There is no hierarchy in SOM, and the topology is static. There is an improved version of SOM, called Growing Hierarchical Self-Organizing Map (GHSOM) (Rauber, Merkl, and Dittenbach 2002). GHSOM introduces hierarchy to SOMs. GHSOM topology is dynamic, and the topology grows with new input data. GHSOM also addresses the problem of SOM nodes with no audio segments that we mention in the Section The Self-Organizing Map. Although GHSOM does not fit Gabora's second creative memory property, we think that musical agents can go beyond human capabilities. With MASOM, we want to move towards the notion of musical agents that listen to music more than a human could (Collins 2017). GHSOM can help to develop musical agents that can be trained on big data of music.

There are many variants of VMM algorithms. Begleiter, El-Yaniv, and Yona (2004) present a comparison of the performance of VMM algorithms on text, molecular biology, and music. The authors show that the performance of a VMM algorithm is context-dependent. For example, LZ-MS performs the best on protein classification whereas LZ-MS and LZ78 perform the worst on predicting English text and symbolic representation of music. Within our knowledge, the comparison of the performance of VMM algorithms on predicting patterns of high-level musical states is still to be done. We plan to compare a set of VMM algorithms in MASOM's system design as a future work.

Acknowledgements

We would like to thank Frederic Bevilacqua from Institut de Recherche et Coordination Acoustique/Musique (IRCAM) for providing us the MAX Sound Box library. Also, we thank Jianyu Fan for his help on the affect estimation model generation. This research was funded by the Natural Sciences and Engineering Research Council of Canada, and Social Sciences and Humanities Research Council of Canada.

References

[2004] Assayag, G., and Dubnov, S. 2004. Using Factor Oracles for Machine Improvisation. *Soft Computing* 8(9):604–610.

[2006] Assayag, G.; Bloch, G.; Chemillier, M.; Cont, A.; and Dubnov, S. 2006. Omax brothers: a dynamic topology of agents for improvization learning. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 125–132. ACM.

[2004] Begleiter, R.; El-Yaniv, R.; and Yona, G. 2004. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research* 22:385–421.

[2013] Bogart, B. D. R., and Pasquier, P. 2013. Context machines: A series of situated and self-organizing artworks. *Leonardo* 46(2):114–122.

[2017] Collins, N. 2017. Towards Machine Musicians Who Have Listened to More Music Than Us: Audio Database-Led Algorithmic Criticism for Automatic Composition and Live Concert Systems. *Computers in Entertainment* 14(3):1–14.

[2007] Dubnov, S.; Assayag, G.; and Cont, A. 2007. Audio Oracle: A New Algorithm for Fast Learning of Audio Structures. ICMA.

[2011] Eerola, T., and Vuoskoski, J. K. 2011. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music* 39(1):18–49.

[2013] Eerola, T., and Vuoskoski, J. K. 2013. A Review of Music and Emotion Studies: Approaches, Emotion Models, and Stimuli. *Music Perception: An Interdisciplinary Journal* 30(3):307–340.

[2010] Eigenfeldt, A., and Pasquier, P. 2010. Real-Time Timbral Organisation: Selecting samples based upon similarity. *Organised Sound* 15(02):159–166.

[1992] Ekman, P. 1992. An argument for basic emotions. *Cognition & Emotion* 6(3):169–200.

[2016] Fan, J.; Thorogood, M.; and Pasquier, P. 2016. Automatic Soundscape Affect Recognition Using A Dimensional Approach. *Journal of the Audio Engineering Society* 64(9):646–653.

[2014] Fried, O.; Jin, Z.; and Oda, R. 2014. AudioQuilt: 2d Arrangements of Audio Samples using Metric Learning and Kernelized Sorting. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Goldsmiths University of London.

[2002] Gabora, L. 2002. Cognitive mechanisms underlying the creative process. In *Proceedings of the 4th conference on Creativity & cognition*, 126–133. ACM.

[2010] Hawryshkewich, A.; Pasquier, P.; and Eigenfeldt, A. 2010. Beatback: A Real-time Interactive Percussion System for Rhythmic Practise and Exploration. *Proceedings of the tenth International Conference on New Interfaces for Musical Expression* 100–105.

[1998] Kohonen, T. 1998. The self-organizing map. *Neurocomputing* 21(13):1–6.

[2013] Lartillot, O.; Cereghetti, D.; Eliard, K.; and Grandjean, D. 2013. A simple, high-yield method for assessing structural novelity. In *Proceedings of the 3rd International Conference on Music & Emotion (ICME3))*.

[2012] Lévy, B.; Bloch, G.; and Assayag, G. 2012. OMaxist dialectics. In *New Interfaces for Musical Expression*, 137–140.

[2007] Livingstone, S. R.; Mhlberger, R.; Brown, A. R.; and Loch, A. 2007. Controlling musical emotionality: an affective computational architecture for influencing musical emotions. *Digital Creativity* 18(1):43–53.

[1997] Moore, B. C. J.; Glasberg, B. R.; and Baer, T. 1997. A Model for the Prediction of Thresholds, Loudness, and Partial Loudness. *Journal of Audio Engineering Society* 45(4):224–240.

[2012] Nika, J., and Chemillier, M. 2012. Improtek: integrating harmonic controls into improvisation in the filiation of OMax. In *International Computer Music Conference (ICMC)*, 180–187.

[2015] Nika, J.; Bouche, D.; Bresson, J.; Chemillier, M.; and Assayag, G. 2015. Guided improvisation as dynamic calls to an offline model. In *Sound and Music Computing (SMC)*.

[1990] Ortony, A., and Turner, T. J. 1990. What's basic about basic emotions? *Psychological review* 97(3):315.

[2003] Pachet, F. 2003. The continuator: Musical interaction with style. *Journal of New Music Research* 32(3):333–341.

[1998] Panksepp, J. 1998. *Affective Neuroscience: The Foundations of Human and Animal Emotions*. Oxford University Press.

[2017] Pasquier, P.; Eigenfeldt, A.; Bown, O.; and Dubnov, S. 2017. An Introduction to Musical Metacreation. *Computers in Entertainment* 14(2):1–14.

[2004] Peeters, G. 2004. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, IRCAM.

[2002] Rauber, A.; Merkl, D.; and Dittenbach, M. 2002. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks* 13(6):1331–1341.

[2004] Roads, C. 2004. *Microsound*. Cambridge, Mass.: The MIT Press.

[1996] Ron, D.; Singer, Y.; and Tishby, N. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine learning* 25(2-3):117–149.

[1980] Russell, J. A. 1980. A circumplex model of affect. *Journal of Personality and Social Psychology* 39(6):1161–1178.

[2014] Smith, B. D., and Deal, W. S. 2014. ML.* Machine Learning Library as a Musical Partner in the Computer-Acoustic Composition Flight. In *the Proceedings of the Joint Conference ICMC14-SMC14*, volume 2014.

[2012] Smith, B. D., and Garnett, G. E. 2012. Unsupervised Play: Machine Learning Toolkit for Max. In *the Proceedings of International Conference on New Interfaces for Musical Expression 2012*.

[2013] Surges, G., and Dubnov, S. 2013. Feature selection and composition using PyOracle. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[2014] Wang, C.-i., and Dubnov, S. 2014. Guided music synthesis with variable markov oracle. In *The 3rd International Workshop on Musical Metacreation, 10th Artificial Intelligence and Interactive Digital Entertainment Conference*.

[2017] Wang, C.-I.; Hsu, J.; and Dubnov, S. 2017. Machine Improvisation with Variable Markov Oracle: Toward Guided and Structured Improvisation. *Computers in Entertainment* 14(3):1–18.

[2009] Wooldridge, M. 2009. *An Introduction to MultiAgent Systems*. John Wiley & Sons.

[2008] Zentner, M.; Grandjean, D.; and Scherer, K. R. 2008. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion* 8(4):494–521.