Constructing Sonic Structures with Visual Roboser

Márcio de Oliveira Costa[1], Jonatas Manzolli[1], Paul Verschure[2]
[1]INúcleo Interdisciplinar de Comunicação Sonora – Universidade Estadual de Campinas (Unicamp) Caixa Postal 6166 – 13091 - 970– Campinas – SP – Brazil
[2]Institute of Neuroinformatics – INI, ETHZ, Switzerland
{marcio,jonatas}@nics.unicamp.br, pfmjv@ini.phys.ethz.ch

Abstract

This paper describes the basic system architecture of Visual Roboser that transforms image sequences into sonic structures. It is an extension of the Roboser Project and it uses images captured from the environment in real time to drive the production of sonic structures. It is based on an architecture that integrates a neural simulator and two JAVA modules: one for image processing and another for sound generation. Possible applications of Visual Roboser include real time composition, multimedia live performances, sonic installations and intelligent interactive rooms such as our latest work name "Ada: the Intelligent Space" (http://www.ada-exhibition.ch).

1. Motivations and Background

In the past six years, we have been working on a novel synthetic interactive composition and performance system called Roboser. It consists of a real-world device, its control software, called IQR421 (Bernardet et all 2002) and a synthetic composition engine, called CurvaSom (Maia et all, 1999). We have explored several aspects of Roboser system such as interactive compositions (Wasserman et all 2000), installations for a general audience in various configurations, such as interactive dance, and communication between humans and large-scale interactive installations (Eng et al., In Press). Here we describe another application of RoBoser that integrates real-time image processing and sound generation. Our goal is to enlarge our understanding of the integration between images and sounds in real time and how this can be used to produce sensible musical structures.

In the literature a series new interface technologies applied to sound generation have been reported (see Paradiso, 1999 for an overview). For instance, the *Very Nervous System* (Rokeby 1998), transforms movements into sonic events using video cameras. Other projects have been described that use robots as part of interactive music systems, for instance EyesWeb (Camurri et al., 2001). EyeWeb integrates movement, music and visual languages in a multimodal perspective.

In all of these approaches the transformation of sensory data to sound is mostly drive by a *Reactive Approach.* Basically, the space of possible interactions is previously labeled with particular sonic reaction. For instance, one specific gesture will trigger one specific sonic reaction.

In contrast to these approaches the Roboser project focused on interactive composition. Our aim is to integrate sensory data from the environment in real time and interface this interpreted sensor data to a composition engine that produces unique emergent musical structures. Instead of having a pre-defined grammar or reactive musical score, our goal is to use the interplay between artifacts and humans to produce new musical structures.

In other words, sonic structures emerge as result of the continuous interaction with a dynamic environment; we call this approach "*Real World Composition (RWC)*".

Starting with the Roboser experience, the Visual Roboser, reported here, emphasizes the exploration of image complexity as a source for emergent sonic structures. We describe how Roboser's capacities -have been extended in the direction of transforming complex and dynamic sensory data, such as real-time video data, into sounds. In this paper we concentrate on the potential of the Java imaging-processing model to support real-time synthetic composition. In particular we investigate whether Java can support the real-time requirements of the RoBoser system. We will describe the system architecture, the current status of the project, as well as the difficulties that we have identified in this approach, and finally we present the future development. Our results show that although the Java2 implementation is not able to fully support real-time processing it can meet the requirements set by the RoBoser architecture.

2. System Architecture

Visual Roboser uses an architecture with two processing levels: a) the neural computing module (NCM) and b) the Java processing module (JPM). The Java level is sub-divided into two components: image processing and sound production as shown in **Figure 1**.

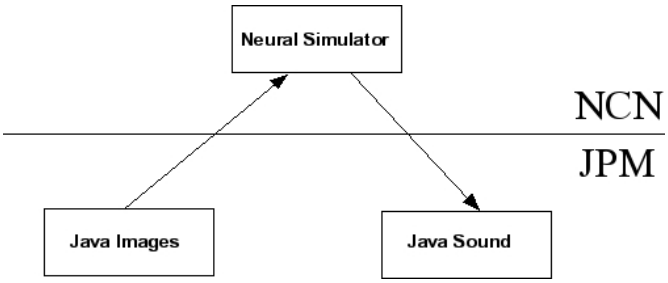

Figure 1. Visual Roboser's system architecture. A Java image processing agent presents high-level image properties to a neuronal simulator, IQR421. The states of the neuronal control system are subsequently projected onto a Java composition agent.

The neural simulator used is IQR421, simulating in the current implementation a static network. This networks associates each pixel from one image (frame) to a single cell, which means for the moment for a frame of 50x50 pixels that 5 populations of 2500 cells are activated representing the color channels (Red, Green, Blue, Yellow) and Luminance. The JPM image-processing component is responsible for acquiring the images and extracting specific parameters such as color, motion information, and luminance. We extract the colors channels (RGB) and luminance of each pixel and send, during the system performance, these values to IQR using a shared memory interface in C. In the NCN each color population is static and their associate fixed weights feed activation to a "voice" population. These populations control the note production. In this setup a total of 25000 cells are mapped linearly to 128 values (the MIDI scale). A winner-take-all mechanism ensures that only the most active cell in every cycle can trigger a sound. **Figure 2** shows the current circuit modeled in IQR421.

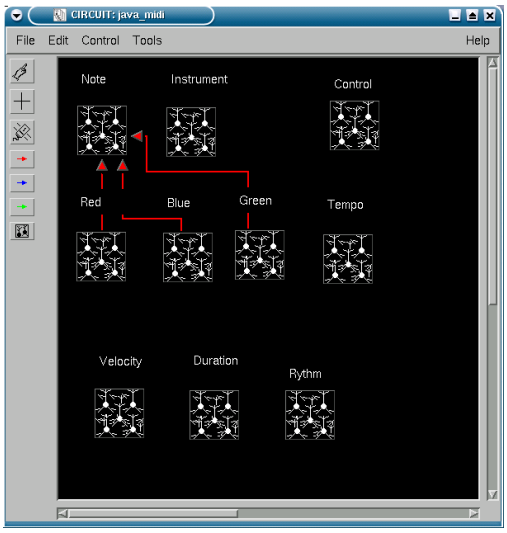

Figure 2: circuit modeled in IQR for sound control and production.

The JPM sound production uses the Java Sound implementation (available since Java 1.3) to generate MIDI in real-time using a custom developed sound toolbox (Costa & Manzolli 2001). Again the link between IQR and this Java module is accomplished using native shared memory code. Several populations control the sound generation module; some of these are show in **Figure 2**, (rhythm, velocity, duration and Tempo). The "control" population plays a special role. As the name says, it controls the behavior of the Java Sound module by using special cells, where any activity above zero activates a special function. These functions include, enabling/disabling the module and changing notes, velocity and rhythm from "*winner takes all logic*" (*WTA*) to "*threshold logic*" (*Th*) (see Bernardet et all 2002, for details).

3. Project Status

In the current implementation a sequence of movie frames is used to extract the following features: **color channels (RGB), luminance, and local and global motion.** These feed in a neural simulation in IQR where the connections are static and predefined at this stage of research. Our result shows that this approach allows for the real-time transformation of image features into a complex musical composition, in the current status, the system generates a sequence of notes according to the pixel activity in the image, as expected by the neural model.

The main difficulties have been the real time constrains. It's difficult to match Java with real time. There is a proposal for a real time specification for Java (http://www.rtj.org/ and an implementation from Timesys but it supports only java 1.2. We are still using recorded images as frames because the pure Java Version of JMF (Java Media Framework) from sun doesn't support live video capture.

4. Results

We have built two different implementations using a fixed weight network. The first one animates a 50x50 pixels frame sequence using a "by hand" thread scheduling (thread.sleep), in other words a frame in the sequence is changed every time the thread wakes up. The second version uses 110x110 images, and the java.swing.Timer class for

image scheduling. Both versions use the new VolatileImage class (available since Java 1.4) for fast off screen drawing. The frame rate was set to 40 frames per second (which means 1 frame each 25ms).

All the measurements are made on a Pentium IV, (1.6Ghz, 1Gb of DDR memory of 266Mhz and Nvidia GeForce 2 MX DDR video card) running SuSE linux 8.0, and java 1.4.1_02 where all classes are compiled with optimization flag).

We measured the time expended in the image processing stage, in other words the process of reading one frame, extracting the color channel information and sending the data to IQR. Here we want to evaluate the system performance potential for being used in real time applications. Results for both implementations are show in table 1.

Table 1: elapsed time in image extraction for different module implementations

	Start	**Warm up 1 (1mim)**	**Warm up 2 (3mim)**
Version 1	26ms	6,7ms	5,6ms
Version 2	20ms	6,7ms	4,6ms

The “start” time is the time elapsed only a few seconds after the module has started, and “warm up” times are the measurements obtained after the java virtual machine has found the module “hot-spots” and compiled them to native code. We can see by these results that all image extraction takes much less then 25ms, the time required for a new frame to be processed.

The table 2 shows the time required to draw the image on the screen for visualization proposesTable 2: elapsed time in image drawing only

	Start	**Warm up 1 (1mim)**	**Warm up 2 (3mim)**
Version 1	1,2ms	1,2ms	0,1ms
Version 2	1,2ms	0,1ms	0,1ms

Here we see that the time spent in drawing (animating) the frames is very small (some times less than 1ms). These results show that the application is acceptable for real-time requirements witch leaves about 20ms for neural processing in IQR.

5. Conclusion and Future Work

The next steps will be to enhance the image features extraction and test the system with MPEG sequences, as well as to change the network to use dynamic and learned neural weights. This will be done using Distributed Adaptive Control (DAC) (see Verschure & Voegtlin, 1998) to change the weights. In principle we will have colors channels, motion, and luminance going to the DAC process. In addition we are experimenting with live video material and other sensory modalities such as sound and touch.

We have implemented only a reactive musical mapping as described in the neural model. The next steps in this project will be to:

a) Evaluate the system real-time performance for a more complex neural simulation, including the DAC methodology.

b) Enhance the sonic control using several image features as a stimulus for the sound model.

c) Compare the musical results of the implemented reactive model with the next one.

One of the most important musical development will be to adapt neural configuration to musical composition strategies. One possibility for musical construction we are evaluating now is to use the logic for note generation described in section 2. We can see that the *WAT* mode will lead to trajectories of individual note events. On the other hand using the *Th* strategy, the system can generate a great number of simultaneous notes that can be understood as an extension of the concept of voice within sound layers. In this case the system will be generating a "*cloud of notes*". Actually, the necessary classes are been designed to test this hypothesis.

On the other hand, we can see Visual RoBoser as one further step in the application of the RoBoser framework for interactive real-world composition. The results presented here shows that this development is critically dependent on the availability of robust real-time processing techniques. Our results show that the Java2 environment is not yet fully mature in this respect. Despite these technical problems at the level of single modulus the overall distributed architecture behind RoBoser is robust and appropriate for the construction of RWC systems.

References

Kynan Eng, David Klein, Andreas Bäbler, Ulysses Bernardet, Mark Blanchard, Marcio Costa, Tobi Delbrück, Rodney J Douglas, Klaus Hepp, Jonatas Manzolli, Matti Mintz, Fabian Roth, Ueli Rutishauser, Klaus Wassermann, Adrian M Whatley, Aaron Wittmann, Reto Wyss, Paul F M J Verschure (In Press) "Design for a brain revisited: The neuromorphic design and functionality of the interactive space Ada". Reviews in the Neurosciences

Maia Jr. A., Manzolli, J., do Valle, R. and Pereira, N.S. L. 1999. "A Computer Environment for Polymodal Music". In the Organised Sound, Cambridge University Press, 02:32-40.

Miranda, E.R. 2002. "Emergent Sound Repertoires in Virtual Societies". In the Computer Music Journal, 26:2, pp. 77-90.

Miranda, E. R., (2003). "On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?", Leonardo, Vol. 36, No. 1,pp. 55-59.

Moroni, A., Manzolli, J., Von Zuben, F. J. and Gudwin, R 2000. "Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition". In the Leonardo Music Journal, Vol. 10, pp. 49-54.

Paradiso, J. 1997. Electronic Music Interfaces: New Ways to Play. In IEEE Spectrum Magazine, 34:12:18-30 1997.

Bernardet U., Blanchard, M. & Verschure, P. 2002. "*IQR: a distributed system for real-time real-world neuronal simulation*", Neurocomputing, 44-46: 1043-1048

Camurri, A., Hashimoto, S., Ricchetti, M., Ricci, A., Suzuki, K., Trocca, R., and Volpe, G. 2000. "EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems". In the Computer Music Journal, 24:1, pp.57-69.

Costa, M. & Manzolli J. 2001. "*Toolbox para Aplicações Musicais na Internet*", VIII Brazilian Symposium of Computer Science and Music, Fortaleza, Brazil, pg: 125-128.

Rokeby, B. 1998. "The construction of experience: Interface as content". Published at Digital Illusions: Entertaining the future with high technology, ed. Association for Computing Machinery, Reading, Ma.: Addison-Wesley.

Verschure, P. F. M. J. and Voegtlin, T. (1998) "A bottom-up approach towards the acquisition retention, and expression of sequential representations: Distributed Adaptive Control III", Neural Networks, 11:1531-1549.

Wasserman, K., M. Blanchard, U. Bernardet, J. Manzolli & P. Verschure 2000. "*ROBOSER – An Automous Interactive Musical Composition System*". In proceedings of the International Computer Music Conference (ICMC – 2000), Berlin, 531-534.