Sound Synthesis from Real-Time Video Images

Roger B. Dannenberg and Tom Neuendorffer

School of Computer Science, Carnegie Mellon University
email: dannenberg@cs.cmu.edu, *web:* www.cs.cmu.edu/~rbd

Abstract

Digital video offers an interesting source of control information for musical applications. A novel synthesis technique is introduced where digital video controls sound spectra in real time. Light intensity modulates the amplitudes of 32 harmonics in each of several synthesized "voices." Problems addressed include how to map from video to sound, dealing with global variations in light level, dealing with low frame rates of video relative to high sample rates of audio, and overall system implementation. In one application, images of light reflected from a shallow pool of water are used to control sound, offering a rich tactile interface to sound synthesis.

1 Introduction

The connection between images, light, and sound has been made countless times. This work uses video to control time-varying spectra, creating an interesting timbre in constant "motion" as the image changes. We describe a working system that controls the time-varying spectra of several voices using live video.

Section 2 describes related work. Section 3 presents the new synthesis method, and Section 4 describes the implementation, which uses off-the-shelf hardware. Section 5 presents ideas for future work and our conclusions.

2 Related Work

Fred Collopy (2003) has written an excellent annotated bibliography, and quoting from his web-site, "the fine art of playing images in the way the musicians play with sound … has gone by a variety of names—visual music, color music, audio-visual-music, motion graphics, synchromy, and lumia." Often, images are derived from sounds, depicting sonic gestures with visual ones. Disney's "Fantasia" is a popular example. In other work, sound is derived from images. For example, Meijer (1992) created a system for the blind in which video images are mapped to audio. Many composers have been inspired by paintings, such as William Kraft in his "Kandinsky Variations." In the computer music domain, MetaSynth (U & I Software, 2003) uses images to control note parameters and Kieren (2003) uses a proprietary image-to-sound conversion system. Penrose's (1992) Hyperupic system converts still images to audio.

Video has been used by many composers as a sensing device, allowing movement to control music. Examples include Rokeby's Very Nervous System (Rokeby, 1998) and some of Winkler's dance and installation pieces (Winkler, 1998). STEIM's Big Eye software (Demeyer, 1996) has enabled many composers to incorporate video sensing into their work. Unlike many of these efforts, our approach uses video to control spectral variation within notes rather than to trigger notes or select pitches.

There are a variety of related synthesis techniques that use one- and two-dimensional data, not necessarily video, as part of the synthesis algorithm. In terrain synthesis, as described and implemented by Rich Gold (Bischoff, Gold, & Horton, 1978), a real or imaginary terrain is scanned in a closed path, and movement of the path causes variations in the generated waveform. Borgonovo and Haus (1985) describe a similar system in which the terrain and path are generated by various mathematical formulas. Scanned synthesis (Boulanger, Smaragdis, & Ffitch, 2000; Verplank, Mathews, & Shaw, 2000) allows the terrain (sometimes in just one dimension) to vary or vibrate at a very low rate, causing interesting variations in the audio waveform.

All of these ideas served as inspiration for our work, which was first described by Dannenberg, et al. (Dannenberg, Bernstein, Zeglin, & Neuendorffer, 2003). While the earlier paper describes an artistic application of the technique, this paper provides complete detail about the implementation.

3 Sound Synthesis from Video

A natural extension to Gold's technique uses video intensity as a source of "terrain" for synthesis. Imagine moving a photocell in a circle over the video image. If the circular path is completed at audio rates (faster than 20 cycles per second), then a quasi-periodic waveform will be generated by the photocell, giving rise to a perceptible pitch. As the image changes, the generated waveform will also change.

We decided to use images from moving water to drive our synthesis engine. Water exhibits interesting wave motion and has obvious parallels to the vibrating shapes in scanned synthesis. To capture water motion, we constructed a shallow pool of water using a wooden frame lined with plastic. We determined that interesting images can be obtained either by shining a light on the water at a shallow angle of incidence in which case light reflects from the water surface, or by using a greater angle in which case light refracts through the water and is reflected from silvered mylar underneath the plastic pool liner. Reflections from the pool can be observed by placing a screen behind the pool (see Figure 1).

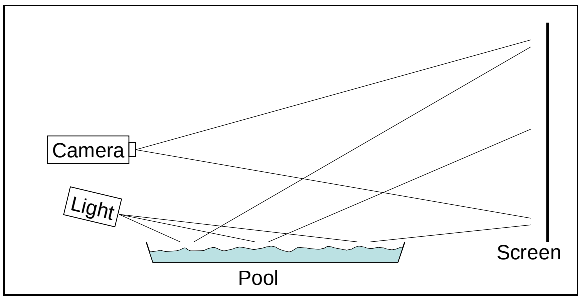

Figure 1. Light reflected from water forms moving images on a screen. These are visible to the audience and captured by a video camera interfaced to a computer.

3.1 Time Domain vs. Spectral Domain

Using video captured from this setup, we determined that time-domain interpolation was not as interesting as we had hoped. One would expect a decrease in high-frequency sound energy as the water waves dissipate. In practice, the high frequencies, represented by small ripples, continue to reflect back and forth for many seconds. At least with water images as an input, the sound is "buzzy": full of high frequencies and lacking in low harmonics. Even though there was obvious wave motion in the image, the generated sound was relatively static, and the connection to the image was not at all obvious. We decided to try a different approach.

Captured video shows that the main image movement is in the form of horizontal bands of light moving vertically (See Figure 2.), due to the position of the light. It occurred to us that time variations along the vertical axis might make an interesting time-varying spectrum. This is the basis for the synthesis technique.

3.2 Computing Time-Varying Spectra

The video images used in this work are 256 pixels square (the power of two is advantageous for texture mapping when the data is also used for animation; the dimensions are not critical for audio synthesis.) A vertical strip 8 pixels wide and 256 pixels high is used to compute each spectrum. For example, we generate three voices using the three strips shown at the bottom of Figure 2. To reduce the noise inherent in video data, each 8-by-8 block of pixels is averaged to a single value, for a total of 32 floating point numbers per strip. (See Figure 3.) These numbers represent the amplitudes of 32 harmonics.

Absolute light levels are difficult to control, so the video processing includes a simple automatic gain control mechanism: the vectors are low-pass filtered in time to obtain an estimate of local average light intensity. The difference between the current vector and this local average estimate is used for synthesis. This will result in some negative amplitudes, and these are replaced by zero.

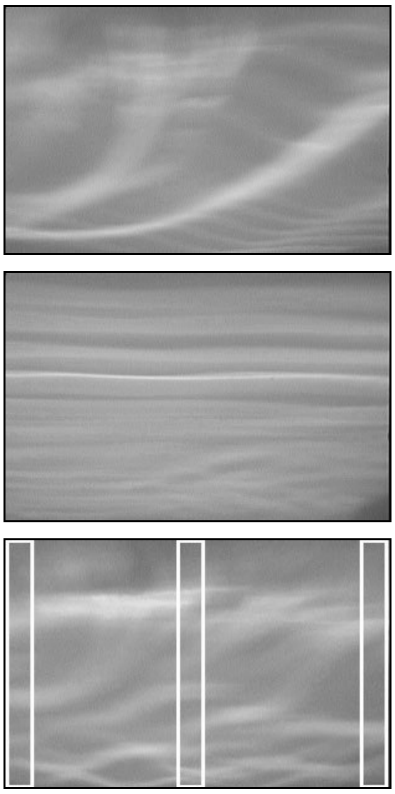
Figure 2. Light reflected from water, captured by digital video. The vertical strips in the bottom picture show which video is used to compute audio spectra.

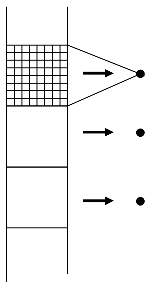
Figure 3. Each 8x8 square of pixels is averaged to obtain one value, resulting in a 32-element vector.

3.3 From Spectra to Sound

Although it might be possible to synthesize using an Inverse Fourier Transform, there are well-known problems associated with discontinuities between successive frames. Also, video frames arrive at a relatively low rate (15 to 30 frames per second) that is not synchronous with audio. Therefore, we use the video data to determine a waveform and then use spectral interpolation synthesis to generate sound.

Spectral interpolation synthesis (Dannenberg, Serra, & Rubine, 1990) was originally developed to model the time-varying spectra of acoustic instruments (Dannenberg & Derenyi, 1998). It works by interpolating between waveform tables that store single periods of the waveform. Interpolation results in a smoothly varying spectrum at a low computation cost because, typically, each waveform table is read for many periods.

Specifically, there is a linear cross-fade from table 1 to table 2, then from table 2 to table 3, etc. Figure 4 illustrates the spectral interpolation algorithm, and Figure 5 shows how amplitude functions are coordinated with waveform changes. In our implementation, we read a video frame every 40ms, so each waveform table will be used for many fundamental periods. Since video frames are not synchronized with audio, special attention must be paid to synchronization. Each cross-fade must not begin until (1) the previous cross-fade finishes (so that interpolation is always between just two waveforms), and (2) the next waveform has been computed from new spectral data (so that there is something to interpolate to). Even if a new waveform is unavailable, the synthesis algorithm continues to generate a tone, and the delay is inaudible. In Figure 5, observe that the cross-fade from wavetable 5 is delayed slightly until wavetable 6 is available. Alternatively, we could sample-and-hold the video data, synchronizing it to the audio, but that would add undesirable latency.

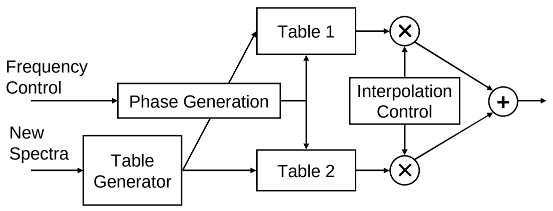

Figure 4. Spectral Interpolation Synthesis algorithm.

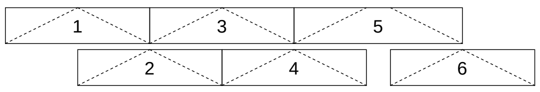

Figure 5. Coordination of waveforms (numbered in sequence) with amplitude control (dashed lines). Waveform 6 arrives later than expected.

Another important point is that only one phase is computed for both tables (again, see Figure 4). If the phases of corresponding harmonics in the two tables are equal, then linear interpolation of the table data is equivalent to an interpolation of their spectra. Notice also that the fundamental frequency is independent of the spectrum, allowing pitch decisions to be made independently from the spectral control.

4 Implementation

An implementation was created for the first author's composition, "The Watercourse Way," in which a dancer creates waves in a pool of water (Figure 1) to control the synthesized sound. Video is "grabbed" using a BTTV8x8 series chipset on a commercial video capture card in a Linux PC, which does all of the sound generation in software. Aura (Dannenberg & Lageweg, 2001) was extended with new objects to perform all video and audio processing. The current implementation generates three voices in stereo at a 44.1kHz sampling rate, and the entire computation uses about 10% of a 2.4GHz Pentium-4 processor.

The video processing consists of reading a frame of video, computing three vectors of 32 harmonic amplitudes, and sending these via Aura messages to software instruments running in a high priority audio synthesis thread.

The audio computation is implemented in C++ and manages wavetable construction, management of wavetables, interpolation control, actual sample computation, amplitude envelopes and stereo panning. Many parameters can be controlled via Aura messages, allowing other parts of the system to control fundamental pitch, turn "notes" on and off, and control panning.

The C++ instrument computes 512-sample tables containing 32 harmonics, which are derived from the video as described above. At any given moment, there are at most three tables: two tables are involved in a cross-fade while the third is computed based on the next spectrum. An even simpler approach would just wait until the next wave table is needed, then compute the table from the video data. This would cause a large computational demand (512 samples × 32 harmonics = 16K samples) that is undesirable in a real-time system. To avoid this, we spread the computation over time, using the third wave table to hold the intermediate data until it is ready. In the current system, audio is computed in blocks of 32 samples, so every time we compute a block of output samples, we also add one harmonic to the third wave table. After 32 blocks of 32 samples (32×32/44100= 23ms), the new table is ready. Even when the new table is ready, the instrument must still wait until the interpolation factor ramps to its endpoint. Then we swap tables and begin interpolating to the new one. This leaves the previous table free to use in constructing the next wave table.

The latest version of Aura runs under Linux and benefits enormously from the human and software resources at PlanetCCRMA (Lopez-Lezcano, 2002), which provides low-latency patches as well as a number of critical device drivers for work in computer music, animation, and video.

5 Discussion and Conclusion

In "The Watercourse Way," three of these Aura instruments are allocated and take spectra from three different parts of the image. Because the water's general texture is similar at all three locations, the general behavior of the three sounds is similar, but each is unique. Algorithmic composition techniques are used to select pitches, but the instruments could be driven easily by MIDI or other data.

The result is a very interesting sound. As water waves travel up and down the screen, the listener hears a sweep through the harmonics, somewhat like a swept filter or a flanging effect. With multiple waves, this gives the sound a very animated character that has a very strong correlation with the moving water image, which was the main goal.

There is no reason to limit input images to light reflected from water. Any motion will create spectral changes, and moving one's hand in front of the camera gives a very pleasing control over spectra, as if one can touch the harmonics themselves. Another amusing effect is to cover the camera lens, creating silence. Removing the cover creates sound, as if one has just let the sound escape from inside the camera.

This work suggests many possibilities for the future. First, there are many choices of synthesis algorithms. One could also use video to control filters to process live audio. The mapping from vertical position to harmonic number gives fairly intuitive and certainly interesting results; by analogy, one might look for other sound processes that are naturally controlled by a one- or two-dimensional vector.

The sounds produced in this work have a distinctive, scintillating "analog" or "filter-sweep" sound. One could add additional controls to limit or shape the overall spectrum, or one could use the video input to control weights on more natural spectra captured from acoustic instruments, for example.

Overall, like many other artists, we find the control information available through video to be very interesting and full of possibilities. This work is fairly original in that is focuses on using video for time-varying control of audio processing as opposed to triggering events or gating sounds on and off. With current laptop and desktop computers, it is relatively simple to obtain time-varying controls from video. We hope this work will encourage and enable others to explore many new possibilities.

Short examples of sounds from this work are available at
http://www.cs.cmu.edu/~music/examples.html.
Source code is freely available from the authors.

6 Acknowledgments

Fernando Lopez-Lezcano and PlanetCCRMA provided much-needed operating system support and expertise. "The Watercourse Way" was created in collaboration with Barbara Bernstein and Garth Zeglin, and received its first performance at Connecticut College's Symposium on Art and Technology. Thanks to Dominic Mazzoni for commenting on a draft of this paper. Parts of this project were supported by NSF Award #0085945 and by the Pennsylvania Partners in the Arts program of the Pennsylvania Council on the Arts, a state agency.

References

Bischoff, J., Gold, R., & Horton, J. (1978). "Music for an Interactive Network of Microcomputers." *Computer Music Journal, 2*(3), 24-29.

Borgonovo, A., & Haus, G. (1985). "Musical Sound Synthesis by Means of Two-Variable Functions: Experimental Criteria and Results." *Proceedings of the International Computer Music Conference 1984*. San Francisco: International Computer Music Association, pp. 35-42.

Boulanger, R., Smaragdis, P., & Ffitch, J. (2000). "Scanned Synthesis: An Introduction and Demonstration of a New Synthesis and Signal Processing Technique." *Proceedings of the 2000 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 372-375.

Collopy, F. (2003). Lumia. http://rhythmiclight.com.

Dannenberg, R. B., Bernstein, B., Zeglin, G., & Neuendorffer, T. (2003). "Sound Synthesis from Video, Wearable Lights, and 'The Watercourse Way'." *Proceedings: The Ninth Biennial Symposium on Arts and Technology*. Connecticut College, pp. 38-44.

Dannenberg, R. B., & Derenyi, I. (1998). "Combining Instrument and Performance Models for High-Quality Music Synthesis." *Journal of New Music Research, 27*(3), 211-238.

Dannenberg, R. B., & Lageweg, P. v. d. (2001). "A System Supporting Flexible Distributed Real-Time Music Processing." *Proceedings of the 2001 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 267-270.

Dannenberg, R. B., Serra, M.-H., & Rubine, D. (1990). "Analysis and Synthesis of Tones by Spectral Interpolation." *Journal of the Audio Engineering Society, 38*(3), 111-128.

Demeyer, T. (1996). Bigeye (Version 1.10). STEIM. http://www.steim.org/steim/bigeye.html.

Kieren, M. E. (2003). Image-to-sound conversion process. Draemgate. http://www.draemgate.com.

Lopez-Lezcano, F. (2002). "The Planet CCRMA Software Collection." *Proceedings of the 2002 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 138-141.

Meijer, P. B. L. (1992). "An Experimental System for Auditory Image Representations." *IEEE Transactions on Biomedical Engineering, 39*(2), 112-121. (Reprinted in the 1993 IMIA Yearbook of Medical Informatics, pp. 291-300.) http://www.seeingwithsound.com/voicebme.html.

Penrose, C. (1992). Hyperupic. http://www.music.princeton.edu/winham/PPSK/hyper.html.

Rokeby, D. (1998). Construction of Experience. In J. Clark Dodsworth (Ed.), *Digital Illusion: Entertaining the Future with High Technology*. New York: ACM Press.

U & I Software. (2003). MetaSynth. http://www.uisoftware.com.

Verplank, B., Mathews, M., & Shaw, R. (2000). "Scanned Synthesis." *Proceedings of the 2000 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 368-371.

Winkler, T. (1998). "Motion-Sensing Music: Artistic and Technical Challenges." *Proceedings of the 1998 International Computer Music Conference*. San Francisco: International Computer Music Association.