A New Control Paradigm: Software-Based Gesture Analysis for Music

Ben Nevile
University of Victoria
bbn@saoul.ca

Peter Driessen
University of Victoria
peter@ece.uvic.ca

W. A. Schloss
University of Victoria
aschloss@finearts.uvic.ca

***Abstract*— We present a flexible and inexpensive system for analyzing gesture data in a computer. Several basic data reductions are introduced, and the tradeoff between latency and reliability is discussed. An example of the application of this technology to musical performance is presented via the example of the radio drum, a musical controller that produces eight channels of gestural data. The advantages of analyzing this data stream in software are exposed, and future applications of the technology are presented.**

I. Introduction

Technologically adept artists explore forms of expression impossible with historically successful media such as canvas, film, tape or presentations of theatre and dance. For many years digital technology has been used by musicians to shape and synthesize sound, and indeed the considerable power of today's microprocessors, plummeting cost of mass data storage, and flexible scope of audio software has made the computer an indispensable tool. Unfortunately, using the personal computer in a real-time music performance can be problematic. If what is desired is to actually *play* the machine in the sense of a traditional instrument, the interfaces currently available to control the sound synthesis are unsatisfactory. For a performer to be able to use our technology in a concert performance we demand that our interface have the following characteristics:

Low Latency A very small delay between gesture and resulting sound is necessary to produce the kind of rapid sonic feedback that a musician receives from an acoustic instrument. We strive to satisfy the 10ms low latency criterion established for satisfactory reactive performance systems[1].

Dynamic Range Musicians should not be quantized. Although the traditional pianist is restricted to a discrete domain of pitches, she enjoys a continuous range of velocity when hitting the keys. We want to design instruments that make it possible to express the full range of human emotion.

Reliability A performer must have confidence in her instrument. The instrument must not fail in the middle of a concert - no more often than a guitar string would snap, or a drum stick break - and it must respond predictably for the performer to be able to develop a deep virtuosity.

II. Creating a Flow of Meaningless Numbers

Most interfaces for electronic music, such as a MIDI keyboard, consist of analog circuitry to sense the gesture, logic to decode the gesture into meaningful musical events, and then hardware ports that shuttle this reduced event information into the computer. In *Communicating with Meaningless Numbers*[2] David Zicarelli points out that the performer-instrument communication is better characterized by a continuous engagement of control rather than by a few discrete events. Zicarelli argues that to find better methods of controlling sound synthesis we ought to be transmitting individual data samples that are stripped of all musical context. The stream of these ”meaningless” numbers would then be analyzed and interpreted in software.

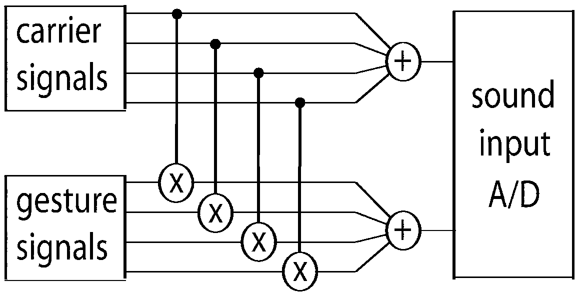

Fig. 1. Block Diagram of Amplitude Modulation Scheme

Accordingly, what we propose is in many ways a less complex approach to communication. Instead of moving data digitally, multiple signals from the gestural interface are amplitude modulated and multiplexed into the stereo sound input of an inexpensive consumer audio card: one half of the stereo input samples the modulated gesture signals and the other samples the carriers, as in Figure 1. Amplitude modulation is necessary because sound cards are AC coupled and do not pass very low frequency signals (below 20 Hz, for example). Since the frequency of muscular contraction has an upper bound of 450 Hz [3] the 22 kHz of standard audio bandwidth provides space to multiplex many channels of gesture data.

Demodulation, analysis and musical mapping of the gesture signals is done in the type of cross-platform modular graphical programming software that new media artists use in a performance context, such as Cycling '74's Max/MSP/Jitter[4] or Miller Puckette's Pure Data[5]. Figure 2 shows an example of a four-channel demodulation patch in Max/MSP. Bandpass filters separate the multiplexed signals, each modulated gesture signal is multiplied by its corresponding carrier signal, and a fourth-order IIR lowpass filter rejects the high-frequency carrier components. Testing indicates that many of today's off the shelf audio cards can move sound into and back out of a computer in less than 300 samples, or roughly 6.8 milliseconds

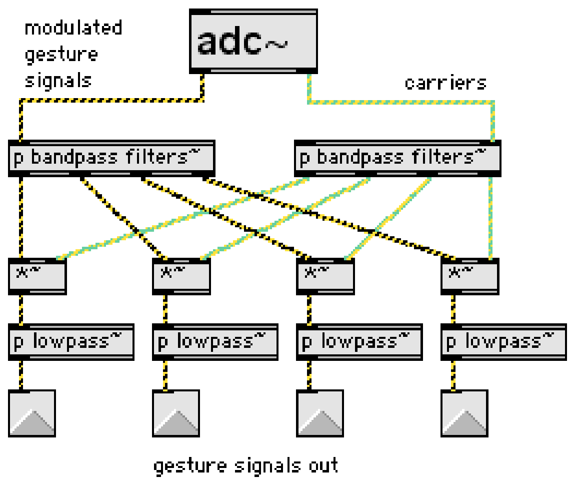

Fig. 2. Four Channel Max/MSP Demodulation Patch

at a sampling rate of 44.1kHz. The impulse response of the IIR filter indicates that it adds approximately 75 samples of group latency, bringing the total to 8.5 ms.

Importing the gesture data through the audio input provides several advantages:

Hardware Simplicity We communicate strictly in the analog domain and have no digital circuitry to engineer into our interface. We take advantage of the hundreds of development hours that the audio card manufacturers have spent creating reliable A/D drivers. As the specifications of inexpensive sound input devices improve - less latency, higher sampling rates and larger bit depth - so does the potential fidelity and responsiveness of our system.

Software Simplicity Because gestures are processed in the same signal chain that renders the audio there is no need for a separate software layer to buffer the data and inject it into the audio signal chain at the appropriate time. Inter-process jitter is eliminated and creation of a fluid, responsive software environment is straightforward.

Flexibility Moving the analysis algorithms from hardware to software can lead to design insight that wouldn't otherwise be gained (see [2]). Complete control freedom is afforded because no decisions have been made on the artist's behalf.

The main drawback to this type of data-processing scheme is an increase in computational load. Fortunately consumer-grade computers now have the horsepower to manage this kind of processing as a small percentage of their total ability. Additionally, opportunities to make the programming efficient by exploiting properties of parallelism and oversampling are abundant in the algorithm design, but because the computational power of accessible computers continues to grow at a staggering pace it is felt that these considerations are secondary in importance.

III. Musically Useful Reduction of the Gesture Data

The presence of noise means that we can only ever deal with the "truth" about these gesture signals in a statistical sense. All processing techniques that can be used to reduce the effect of noise in a signal must take into account the value of multiple samples, and therefore it is unavoidable that precision in measurement or detection of a signal can only be achieved at the cost of delay over and above the latent characteristics of the computer's sound input device.

Position The most straightforward way to reduce the effect of noise in a signal is to simply take the mean of m successive samples: assuming that the actual value of the signal is constant - a fair assumption with our techniques, since our gesture signal is hugely oversampled - the variance in the signal has an inverse relationship with m, whereas the latency inherent in the evaluation increases linearly as m grows larger. The longer we are able to delay our estimation, the closer our determination is likely to be to the actual position.

Derivatives A velocity signal can be estimated from a position signal by subtracting the value of successive samples. Similarly, acceleration can be obtained by calculating the difference between successive estimations of velocity, and higher moments may be estimated by repeating the process. Note however that each transformation to the next higher derivative doubles the variance in the signal due to noise, and so can require considerable smoothing (and hence delay) to achieve reliable results.

Thresholding and Regions Starting from the one-dimensional case, a threshold is some value t that we wish to compare to the signal: is our signal greater than or less than t? It is unfortunately impossible to simultaneously maximize the probability of detecting when the signal is above the threshold and minimize the probability of mistaking noise for signal above the threshold; the best we can do is keep the so-called *false-alarm probability* p_{fa} at a tolerable level. Given an estimation of a stick's position, the variance in the statistic and some assumptions about the distribution of noise (that it is white Gaussian, say) we can calculate a buffer amount b such that if the estimation of the position is above $t + b$ the probability of the stick's actual position not being above t is less than p_{fa}. Once again the tradeoff between latency and accuracy is evident: the more samples we consider in our estimation, the less the variance and therefore the smaller b has to be to ensure a probability of failure less than p_{fa}. Since joint probabilities become products of individual probabilities for independent events, this one-dimensional analysis can be extended to segmented regions and multiple dimensions simply by applying similar restrictions for each threshold value and dimension and multiplying the boolean results.

Pulses If we want to identify a pulse or known amplitude variation in the signal, the algorithm which yields maximum sensitivity is the matched filter. The expected pulse shape can be sampled at m points $s_1...s_m$ to be used as the coefficients of an FIR filter, and the peaks of this filter's output will correspond to the times of maximum likelihood that the pulse has been input.

Since the probability of a false alarm due to noise P_n cannot be minimized with the same algorithm that maximizes the probability of detecting a valid pulse P_d, one must maximize

P_d with P_n set to a tolerable level. The optimum Neyman-Pearson processor consists of comparing the output of our matched filter $y = \Sigma_{j=1}^{m} s_j x_j$ to a threshold level d determined by our choice of P_n. Under the assumption of Gaussian noise with variance σ_n^2, very small values of P_n have approximately the following relation to d[6]:

$$P_n \approx \frac{e^{-\left(\frac{d}{\sqrt{2}\sigma_y}\right)^2} \sqrt{2}\sigma_y}{2\sqrt{\pi} d} \tag{1}$$

where $\sigma_y = \left(\sqrt{\Sigma_{j=1}^{m} s_j^2}\right) \sigma_n$ is the standard deviation of the noise in the output of the matched filter. For $P_n = 10^{-10}$, which at a sampling rate of 44.1 kHz corresponds to a false alarm once every 63 hours, $d = 6.35\sigma_y$. We can then express the detectability of a pulse as a function of the effective signal-to-noise ratio, given by the ratio of its energy E to the noise spectral density n_0:

$$P_d = \frac{1}{2}\left[1 - \text{erf}\left(\frac{d}{\sqrt{2}\sigma_y} - \sqrt{\frac{E}{n_0}}\right)\right] \tag{2}$$

Parallel Operations Imagine that we wish to identify a multi-part gesture that requires input from more than one interpretive process: for instance, we may wish to trigger a noise when a sensor's velocity exceeds a certain value but only while a different sensor is in a particular region. The different processes required to extract the different features each has its own effective latency. Care must be taken when features are extracted in parallel to ensure that the latencies inherent in the interpretive processing are of equal length, otherwise the logic won't be aligned in time. If a particular reliability or accuracy goal is sought a probability analysis of each individual event must be undertaken.

IV. The Radio Drum

As a test case for this technology we consider the radio drum, an instrument created at Bell Labs[7]. The two sticks of the drum are electrically driven with a radio frequency voltage source, and the division of displacement current between the four corners of the flat sensor surface determines the eight output signal strengths. These signals are then transmitted to circuitry that amplitude modulates and sums the signals in the manner of Figure 1, after which they are passed into the computer through the sound card as a stereo pair.

The most straightforward information to extract from the signals sent by the radio drum is the three-dimensional location of a sensor. Through the transformations outlined in [7] the eight channels of data are reduced to two three-channel groups representing the x,y, and z positions of each stick. Estimating the actual position of the sticks is then as straightforward as a simple averaging scheme over m samples.

Since this is a *drum* it seems natural that we would want to detect when the foam surface has been hit. In a previous paper[8] it was proposed that an FIR filter with 2048 taps be used to smooth the signal, and then a simple first derivative peak picker be used to identify the extremities of the signal.

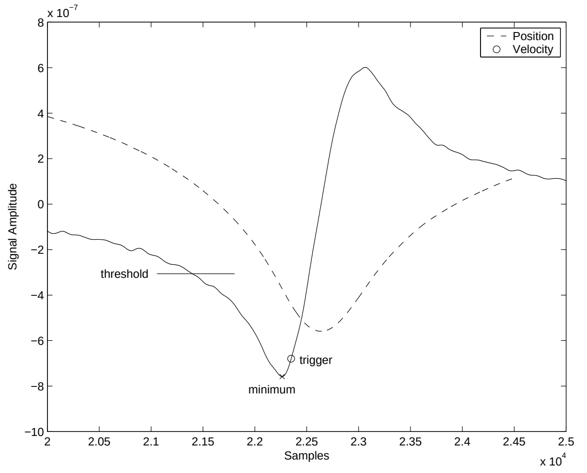

Fig. 3. Position and Velocity Signals of a Radio Drum Hit

Although the system worked well in theory, the 2048 tap FIR introduced an unacceptably large latency of 23.2ms at a sampling rate of 44.1kHz.

We present in this paper a novel technique inspired by work in radar pulse detection [9]. Figure 3 shows the signal corresponding to a drum hit and its first derivative on the same graph with their amplitudes scaled to the same range to facilitate comparison. The crucial idea is that the downward peak of the first derivative happens *before* that of the filtered gesture signal; by tuning our algorithms to locate this point in the signal instead of the later "real" peak we gain on average more than 6.8ms in our battle against latency. In fact, we suspect that this extrema of the first derivative is a more appropriate point to choose to trigger sound synthesis, since it likely represents the point of first contact with the surface of the drum.

We first smooth the signal with 256-points of a matched filter tuned to the expected shape of a hit, after which we subtract successive samples to determine the velocity signal v. We use the first derivative instead of a ratio (see [9]) because of the derivative's preferable signal-to-noise characteristics in the drum's range of signals, and regain the amplitude-independency with a variable trigger threshold: when a minimum v_{min} is encountered in v the trigger $t = r_t v_{min}$ is set using a factor $0 \leq r_t \leq 1$. When the stick slows down enough for the velocity to increase above t the algorithm communicates a *hit event* to the synthesis engine. Setting the ratio factor $r_t < 1$ helps avoid false alarms due to the inevitable small noise fluctuations in v, which has twice the variance of our position signal z. A value of $r_t = 0.9$ was used in Figure 3.

After a hit has been output by the algorithm, a new trigger will not be set until $v > 0$ - ie, the stick must move upwards before another hit will be recorded. Furthermore, a trigger point will not be set unless v has at some point in the descent reached a threshold level $vThreshold$ which we set significantly above the noise floor of the stationary velocity signal. Finally, the estimated position of the stick must be below a threshold level at the surface of the drum to avoid

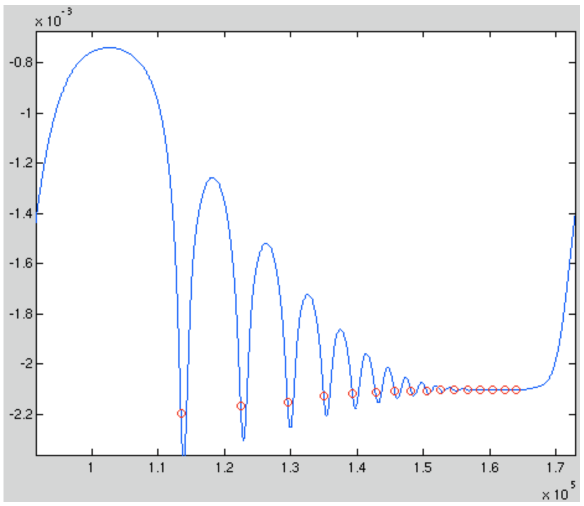

Fig. 4. Filtered Signal With Detected Hits

triggering on a reversal of direction in mid-air. A pseudocode listing of the peak detection algorithm follows:

```
for each sample z(n)
  v(n)=z(n)-z(n-1)
  a(n)=v(n)-v(n-1)
  if (v(n)>0)
    vHasGoneUp=1
  elseif ((v(n)<vThreshold)&&(vHasGoneUp==1))
    vFast=1
  elseif (((a(n)>0)&&(a(n-1)<=0))&&(vFast==1))
    t=v(n)*rt
  elseif ((t!=0)&&(v(n)>t)&&(v(n)<0))
    if (z(n)<physicalThreshold)
      communicate a hit event
      vFast=0,  vHasGoneUp=0,  t=0
```

Figure 4, a plot of the detected hits over the filtered gesture signal of a drum roll of decaying magnitude, makes it clear that the sensitivity of the algorithm is outstanding. Just as important, the latency is very close to being acceptable: the algorithm presented here introduces a further 128+1 sample delay from the matched filter and the differentiation, which when added to the 8.5ms of input and IIR delay makes a total of 11.4ms. However we can consider our derivative analysis as saving us at least 6.8ms from detecting the peaks of the position signal. The other useful parameters of the hit - the force and x, y positions - are captured when the trigger is set and sent to the synthesis algorithm once a hit has been triggered. Future work may include experimenting with other interesting musical analyses of the signal, such as detecting ”hits” in mid-air, finally turning *air drumming* into the practical art form it deserves to be.

V. Conclusions

Figure 5 shows the Max/MSP/Jitter patch that acts as console for one stick of the radio drum, the x, y and z signals waiting to be used in an artistic way. The choice of which of the thousands of synthesis algorithms to drive with these control signals is certainly in the domain of the musician, but our work begs the question: where does the instrument

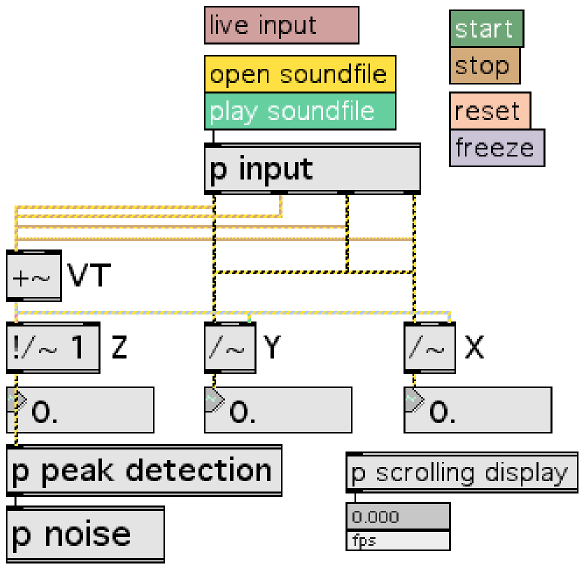

Fig. 5. Max/MSP/Jitter Radio Drum Console

design end and the art begin? It stands to reason that as deep, expressive interfaces develop, the digital processes that shape the control signals will be as important as those that generate the sounds. Future work ought to move towards a generalized set of powerful, high-level signal processing abstractions that make it convenient for the musician to experiment with the new control parameters available to him.

Finally, it is worth pointing out that a system resolved, reliable and flexible enough for a professional musician will certainly find use in all manner of difficult interface problems. Future work in assistive technologies will help us to understand what characteristics of an interface are important, and how to strike a balance between a system that's intuitive yet rich in control possibilities.

References

[1] A. Freed, A. Chaudhary, and B. Davila, “Operating systems latency measurement annd analysis for sound synthesis and processing applications,” 1997.

[2] D. Zicarelli, “Communicating with meaningless numbers,” *Computer Music Journal*, vol. 15, no. 4, pp. 74–77, 1991.

[3] T. M. Nakra, “Searching for meaning in gestural data,” *Trends in Gestural Control of Music*, 2000.

[4] [Online]. Available: http://www.cycling74.com

[5] [Online]. Available: http://crca.ucsd.edu/~msp/software.html

[6] M. Schwarz and L. Shaw, *Signal Processing: Discrete Spectral Analysis, Detection and Estimation*. Dubuque, Iowa: McGraw Hill, 1975.

[7] R. Boie, L. W. Ruedisueli, and E. R. Wagner, “Gesture sensing via capacitive moments,” internal memo at AT&T Bell Labs.

[8] W. A. Schloss and P. Driessen, “New algorithms and technology for analyzing gestural data,” *Pacific Rim Conference*, 2001.

[9] K. C. Ho, Y. T. Chan, and R. J. Inkol, “Pulse arrival time estimation based on pulse sample ratios,” *IEE Proc.-Radar*, pp. 153–157, August 1995.