Sound Bytes

Electronic music gains a human touch

Somewhere, off in the distance, a gentle swishing begins to build, the sound of sand tumbling over a snare drum. Suddenly a sharp tinkling breaks in: tiny mallets striking crystal wineglasses. The blows fall more frantically, the clinking pitches higher—

While listening to composer Tod Machover's piece "Bug-Mudra," it is not difficult to keep in mind that much of the music is being generated by a computer. But watch Machover conduct the work on stage, and it is the integration of human and electronic performers that is unforgettable.

With his right hand, Machover directs a group of musicians with the familiar swings of a conductor. Meanwhile his left hand, encased in an enormous cybernetic "data glove" covered with sensors, directs a Macintosh II computer that controls a section of synthesizers. "My goal is to build things that make music become more creative, not just prosthetic devices," declares Machover, who is a professor at the Media Laboratory at the Massachusetts Institute of Technology. He currently records with Bridge Records in New York City.

Much electronic music continues to be developed precisely where it started—in the laboratories of computer scientists. But those who have spent years creating computer music, including Machover, are determined to keep people in the act. To do so, they are exploiting complex algorithms and new hardware to enhance the music performances of humans.

"Many music appreciators don't have the ability to play music but love it," says Max V. Mathews, a professor at Stanford University's Center for Computer Research in Music and Acoustics. More than 20 years ago Mathews helped to spark the synthetic music revolution with algorithms he devised at Bell Laboratories. Now he hopes to put music back into the hands of the listeners with his "radio baton." In many ways, it is a simpler version of the data glove used by Machover.

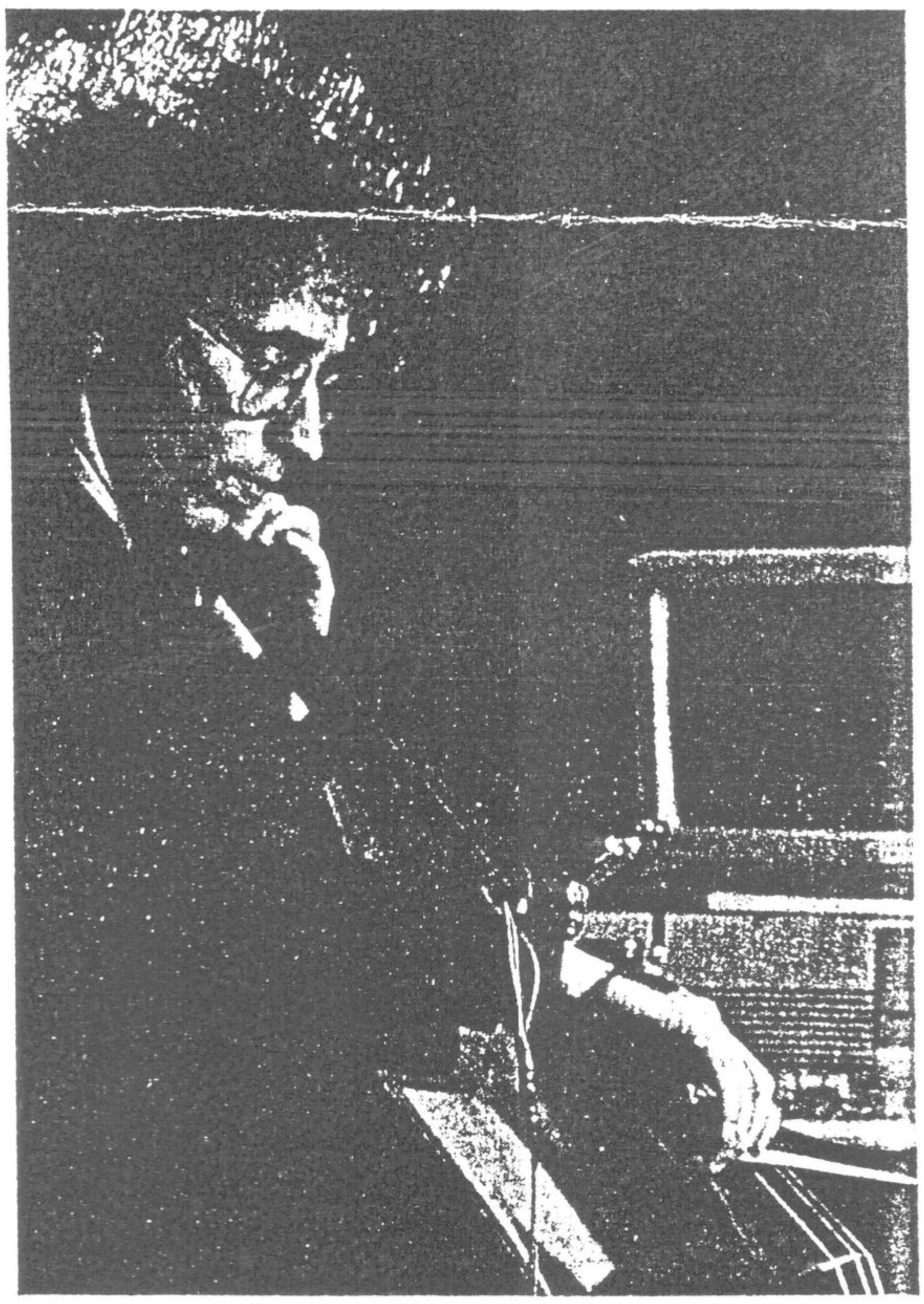
The radio baton is packed with several simple transmitters that send signals to an array of receivers that track the motion of the baton in three dimensions and communicate the data to a computer-simulated orchestra. A gesture to the left may enhance a bass line; a motion downward may speed up the tempo. "Instead of passively listening, people will be able to buy scores and conduct their own interpretations," Mathews predicts.

Others are trying to make computers more suitable performers in ensembles. "Our focus is to make the computer into a live instrument," explains Miller Puckette, a researcher at the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) in Paris.

Puckette and his colleagues have been developing algorithms that enable a computer to track the notes being played by a nearby live instrument and calculate its place in the music score. "Finding the pitch is a terrible analysis problem," he says. IRCAM is also building algorithms that let the computer follow the leads from other instruments. After picking notes from an ongoing performance, other, established algorithms will generate new combinations—chords or arpeggios, for example—around them.

Several investigators are developing "listening assistants"—computer programs, including neural networks, that follow music as it is being played and learn ways of improvising from the underlying structure of the piece. These days, David Wessel, who directs the Center for New Music and Audio Technologies at the University of California at Berkeley, is often found playing jazz-inspired improvisations with two musicians and a computer-based listening assistant. "The assistant takes fragments of what they're playing, elaborates on them and reinjects them in an immediate way into the performance," Wessel says.

Similarly, the "hyperinstruments" being built by Machover and other researchers at M.I.T. interpret an ongoing performance in novel ways. Musicians play hyperinstruments much as they would acoustic ones. The electronic hybrids, however, are packed with sensors that pick up every inflection of the performance. The data are the basis for new synthesized sounds that enhance the performer's original expressive intent.



HYPERCELLO, played by Tod Machover, is a hybrid of sensors and instrument that elaborates on the music being played. Photo: Peter Menzel.

So that computers will be regarded as performance instruments in their own right, investigators are also looking for ways to "recapture the intimacy of sound" by incorporating the nuances of acoustic instruments, Wessel says. Push a bow across the string of a violin, for instance, and the note produced will be a rich collection of sounds: the initial attack followed by a crescendo that fades to a subtle buzz. Computer-driven synthesizers, in contrast, have typically produced more uniform sounds. "The disadvantage is fixed in the hardware," says Carla Scaletti, president of Symbolic Sound Corporation in Champaign, Ill.

Typically, conventional music systems store a collection of digitized sounds—say, all the notes played by a piano—or a specific algorithm that dictates how to combine signals. Call for middle C, and the computer sends the binary description of the note to a synthesizer. What Scaletti and others aim to do is replace that library of digital signals with digital-signal processors that can be easily reprogrammed by a user. In this way, a person could redesign C or build a completely new sound.

So far synthesizers based on digital-signal processors are still expensive and tricky for musicians to use, experts say. But a handful of companies are beginning to develop commercial products. Early this year Symbolic began selling Kyma, a hardware and software system based on as many as nine digital-signal processors. Kyma includes 80 initial "sound objects." With an icon-driven program, a user can create a palette of other sounds by performing various operations on the data. "This is set up for exploring—not for someone under intense time constraints," Scaletti cautions.

Advances in digital-signal processing will "open doors for us to really think about music in different ways," Machover adds. Musicians, in turn, must convince audiences that there is substance and depth behind the novel orchestrations. One opportunity will take place soon: in August cellist Yo-Yo Ma debuts Machover's latest work, a 25-minute piece in three movements for three hypercellos at Tanglewood in Lenox, Mass. The audience's reaction will, of course, be live. *—Elizabeth Corcoran*

FORTRAN Forever

Is it still the language of choice for science?

If you wanted to program a computer for scientific or engineering problems during the 1960s, FORTRAN was the language you used. It was clunky and unforgiving, but it was all there was.

Then computer scientists came up with languages that were easier to use, more elegant and more powerful: C, BASIC, APL, LISP, PASCAL and others. But physicists and engineers kept right on using FORTRAN. They analyzed stresses on bridges and aircraft, modeled the inside of fusion reactors and predicted the evolution of supernovas, all in a language firmly tied to punch cards.

With the advent of multiprocessor computers that break problems into parts and solve each section simultaneously, language designers began whipping up a whole new set of computer dialects. For these parallel processors, they invented SETL, NETL, ACTORS, LINDA and more. But recently a group of software engineers completed a 13-year effort to standardize an older computer language they claim is almost perfectly suited for a wide range of parallel programming tasks.

What's it called? FORTRAN. Well, FORTRAN 90, also known as FORTRAN EXTENDED. (Previous versions were FORTRAN 66, which was standardized in 1966, and FORTRAN 77, which was actually approved in 1978.) The international version is done, and approval of a U.S. standard is expected by the end of June.

The renovated FORTRAN was really designed to make it easier to write scientific or engineering programs. But by a stroke of good fortune, the same features that simplify life for programmers also furnish clear guideposts for running the programs on machines incorporating hundreds or even thousands of processors, says Danny Hillis, the parallel computer designer and founder of Thinking Machines Corporation in Cambridge, Mass.

The most important change in FORTRAN 90 is new instructions that can manipulate arrays of data—for instance, the temperature and wind velocity at every point on a grid covering the Western Hemisphere—as a single unit. A compiler (the computer program that translates FORTRAN statements into binary code) can turn such statements into commands that carry out operations on each element of the array in parallel. In the past, programmers had to write instructions to handle each data item individually.

Geoffrey C. Fox, a computer scientist at Syracuse University, estimates that FORTRAN 90 will be nearly perfect for the parallel programming of about half of all the scientific and engineering computing done today. And many other programs can be recast to fit the kind of parallelism that FORTRAN 90 supports, he says.

The evolution hasn't been easy. Work on updating FORTRAN has been going on for nearly 14 years. Now although a number of upstart computer companies, including Thinking Machines, MasPar, Alliant and AMT, already have compilers that incorporate much of FORTRAN 90, older, more established firms have been dragging their feet, Hillis says.

Moreover, some programmers—the ones whose life FORTRAN 90 was intended to simplify—are not rushing to embrace the improved version. They complain that it bears little resemblance to earlier versions and has features that will be difficult to learn. Fortunately, users can take their time learning the new dialect. Old programs will still run under the new standard, says Jeanne Adams of the National Center for Atmospheric Research in Boulder, Colo., who chairs the FORTRAN 90 standards committee at the American National Standards Institute.

But if they take too long, FORTRAN may change again. Fox and others foresee extending FORTRAN 90 in a few years to increase the range of easy-to-write parallel programs and to put back features removed from early drafts of the language by conservative standard-setters. How many times can the venerable language be updated? Computer-language designer Guy L. Steele of Thinking Machines quotes an unknown scientist: "I don't know what the computer language of the year 2000 will look like, but I know that it will be called FORTRAN." *—Paul Wallich*