Room service, AI-style

By Haym Hirsh
Rutgers University
hirsh@cs.rutgers.edu

Can a room be intelligent? This month's "Trends and Controversies" presents the thoughts and work of four people who not only believe the answer is yes, but are working towards making this happen.

Michael Coen's leadoff essay, "The future of human-computer interaction or how I learned to stop worrying and love my intelligent room," discusses insights gleaned from work at the Intelligent Room project at the MIT AI Lab. An intelligent room should base its actions on an integration of sensing modalities, such as vision and speech, performing tasks (such as speaker tracking) that would be more difficult to perform using any of the modalities in isolation. Interestingly, the lion's share of the effort did not involve work on the technologies of the individual modalities, but rather in understanding how to structure the task so that a range of sensing-modality subsystems can interoperate and be integrated into a seamless whole. Coen also proposes general principles for future builders of intelligent rooms coming out of his own experience, such as "Do not scavenge"—individual components must be designed with an eye to how they will ultimately be used as part of the overall whole, rather than being inherited as a by-product of someone else's goals.

In contrast to the two intelligent rooms built in the MIT AI Lab's quarters in an otherwise (comparatively) mundane office building in Cambridge, Massachusetts, Michael Mozer's essay discusses his experiences making an entire house in Colorado behave intelligently. Whereas much of Coen's efforts center on dealing with the integration of various sensing modalities, Mozer's focus is instead on adaptability, building an intelligence into the various sensors (such as thermometers) and effectors (such as a heating system) so that it can adapt to the preferences of the house residents. How the intelligence can infer the preferences of its residents is one of the important questions that Mozer discusses, such as by learning from any time an inhabitant manually adjusts the settings of lights or thermostats. As Mozer points out, although we may believe that our habits are far from predictable, for the vast majority of the time that we do not even notice, we are very predictable—for example, if we are happy with the environment in the house at one point in time, we are usually happy with it five minutes later.

Mozer's adaptive house began as an existing home, renovated to serve as a testbed for intelligent-room research. But what if you are building a brand new residence from scratch? Moreover, what if you are not constrained by the more typical budgetary limitations of the average American home buyer? Richard Hasha's essay discusses his experiences as chief systems architect for the Gates Estate, Bill Gates' well-publicized 66,000-square-foot new home in Medina, Washington. Hasha's focus is on the development of an underlying architecture that is able to scale up to the demands of building intelligence into a home of this magnitude. To do this, Hasha identifies some of the "plumbing-related work" that must be performed as part of any large-scale effort to build an intelligent home, and describes a distributed-object framework that supports it.

It is not only offices and homes that can be intelligent. James Flanagan's final essay considers some of the functionalities that we would want intelligent rooms with large numbers of people—such as lecture halls and conference venues—to possess. For example, we would like an intelligent lecture hall to recognize automatically who in the audience is asking a question, point a video camera at that person, and position and use a microphone array to filter out sounds coming from elsewhere in the room. Flanagan breaks up the problem into two pieces—identifying the spatial location of the sound source, and then extracting the desired sound signal emanating from that source out of the collection of sounds being received by the microphones. He points out how recent advances, such as in both microphone and digital-signal-processing technologies, are enabling the creation of such intelligent rooms, and the technical advances that have made this possible.

Twenty years ago, John McCarthy asked whether we could ascribe mental qualities to a thermostat (J. McCarthy, "Ascribing Mental Qualities to Machines," *Philosophical Perspectives in Artificial Intelligence*, M. Ringle, ed., Harvester Press, Brighton, Sussex, UK, 1979). Perhaps someday we will be able to ask them ourselves.

—Haym Hirsh

The future of human-computer interaction, or how I learned to stop worrying and love my intelligent room

Michael H. Coen, MIT Artificial Intelligence Lab

Predicting the future is notoriously difficult. Suppose 100 years ago someone suggested that every bedroom in the US would soon have a bell that anyone in the world could ring anytime, day or night. Would you have believed it? Nevertheless, the telephone caught on and has become a technology conspicuous only by its absence.

Now, I find myself in a similar position. Having spent the past four years immersed in the future as part of the MIT AI Lab's Intelligent Room project, I have gained an inkling of what is to come in the next 50. I have taken to heart the advice of Alan Kay, "the best way to predict the future is to invent it." Of course, this is not a solo performance, and the cast of fellow prognosticators (researchers and inventors) who work on similarly futuristic environments such as the intelligent room has grown markedly. [1] It is interesting to note that this cast is equally divided among industrial and academic research labs. There are, I think, two reasons for this:

- intelligent rooms are an incredibly exciting testbed in which to do research (read the rest of this essay to find out why), and
- there is a lot of money to be made. Intelligent rooms promise to have the ubiquity of, well, rooms and the upgradability of PCs—you do the math.

The starting—and I think quite uncontroversial—premise for my research is that computers are not particularly useful. Of course, scientists and engineers adore them, as do a growing cadre of Web surf-

Intelligent Rooms

Figure A shows two sample scenarios with the Intelligent Room that run today, reflecting both our interests and those of our funders.

At play: (Our interests...)
Me: I walk inside my office.
Room: Turns on lights and says, "Good evening, Michael, you have three messages waiting."
Me: I sit down on the couch.
Room: Displays messages on one projected display. Shows a video on the other.
Me: I lie down on the couch and lay still for a while.
Room: "Michael, are you still awake?"
Me: "No, wake me up at 8 am."
Room: Turns off video, dims the lights, closes the drapes. And puts Mozart on softly in the background.

At work: (Our funders'...)
Me: I walk inside my office and say, "Computer, activate the command post."
Room: Shows interactive world map on one display and Web browser on the other. Closes the blinds and drapes. Says, "The command post is activated."
Me: Mark region on map with a laser pointer, "Zoom in here"
Room: Zooms in on the map.
Me: Point at country using laser pointer, "What is this country?"

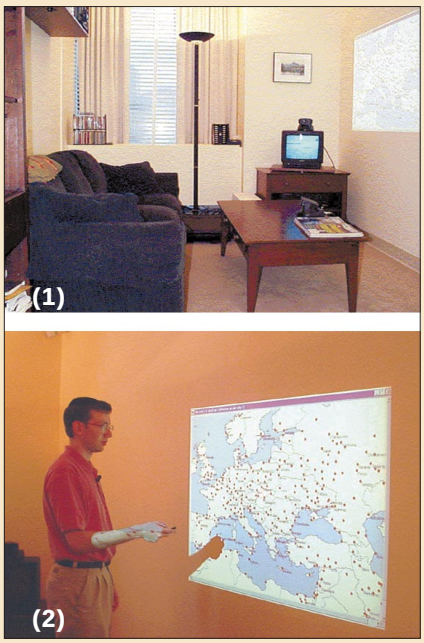


Figure A. Intelligent room scenarios: (1) our interests and (2) our funders' interests.

Room: "You are pointing at Iraq" and displays CIA World Fact Book information about Iraq in Web browser
Me: "I need information; does Iraq have ballistic missiles?"
Room: Displays Iraq's current missile capabilities in the Web browser.

ers, e-mailers, and online socialites. And certainly, computation is a fundamental component of our society's technical, financial, and industrial infrastructures; machines are outstanding data-processing slaves. But in terms of raising our quality of life, computers have a very far road to travel before becoming as essential as the light bulb, indoor plumbing, or pillow.

The reason is obvious. Computers are generally used for things that are computational, such as reading e-mail, and for most of us, the majority of our lives are spent doing noncomputational things, such as taking baths and eating dinner. Most people spend their time in the real world, not in the much-ballyhooed realm of cyberspace, and as a description of the current utility of computation, I propose the following observation: *the value of a computer decreases with the square of your distance from its monitor*. That being said, computation will not become sociologically essential until computers are connected to the human-level events going on in the real world around them—until they are ingrained in and conversant with our ordinary state of affairs. Borrowing once more from Kay, "the computer revolution hasn't happened yet."

In the Intelligent Room project, we are interested in creating spaces in which computation is seamlessly used to enhance ordinary, everyday activities. We want to incorporate computers into the real world by embedding them in regular environments, such as homes and offices, and allow people to interact with them the way they do with other people. The user interfaces of these systems are not menus, mice, and keyboards but instead gesture, speech, affect, context, and movement. Their applications are not word processors and spreadsheets, but smart homes and personal assistants. Instead of making computer interfaces for people, it is of more fundamental value to make people interfaces for computers. The need for doing this is not new; it has simply become more urgent:

> The great creators of technics [i.e., technology], among which you are one of the most successful, have put mankind into a perfectly new situation, to which it has as yet not at all adapted itself.—Albert Einstein, in a tribute to Thomas Edison, 21 Oct. 1929.

It is time for technology to start adapting to us. This might sound trite but only because it is so obviously true.

Sounds great, but how?

We have built two intelligent rooms in our laboratory, where our approach has been to give the rooms cameras for eyes and microphones for ears to make accessible the real-world phenomena occurring within them. A multitude of computer-vision and speech-understanding systems then help interpret human-level phenomena, such as what people are saying and where they are standing. By embedding user interfaces this way, the fact that people, for example, tend to point at what they are speaking about is no longer meaningless from a computational viewpoint, and we can (and have) built systems that make use of this information.

Coupled with their natural interfaces is the expectation that these systems are not only highly interactive—they talk back when spoken to—but more importantly, that they are useful during ordinary activities. They enable tasks historically outside the normal range of human-computer interaction by connecting computers to phenomena (such as someone sneezing or walking into a room) that have traditionally been outside the purview of contemporary user interfaces. Thus, in the future, you can imagine that elderly people's homes would call an ambulance if they saw anyone fall down. Similarly, you can also imagine kitchen cabinets that automatically lock when young children approach them.

The most important factor in making intelligent rooms possible in recent years has been the novel viability of real-time computer vision and speech understanding. AI, and computer science more generally, have experienced something of a renaissance in the past decade, with many research areas blossoming almost entirely due to the sudden and unexpected availability of inexpensive but powerful processors. It is now entirely possible to have literally a dozen computer-vision systems as components of a larger project, the suggestion of which would surely have raised more than a few skeptical eyebrows in the not-too-distant past.

Computer vision and speech understanding are among the preeminent members of a class of research problems known as being *AI-hard*; namely, they are as difficult to

solve as anything else we don't yet know how to do. When we started working on our lab's first Intelligent Room, we expected the vision and natural-language subsystems would require the overwhelming majority of our intellectual effort. What was not initially obvious was the amount of forethought that would be required to integrate the room's myriad subsystems and from them produce a coherent whole. Building a computational system—one that had literally dozens of hardware and software components—that allowed its subsystems to not only interoperate but leverage off one another eventually emerged as our project's chief research problem.[2] In this, we have not been alone; finding some way of managing similar systems and moving data among their components was the foremost difficulty raised at the Intelligent Environments Symposium.[1]

In some regards, our solution to this management crisis has been a bit drastic: we created a new programming environment, called Metaglue, to meet the room's fairly unique computational needs, in which (at last count) the room's 80 software components are distributed among a dozen workstations.[3] We also formulated some general principles for creating intelligent rooms, which we now adhere to with religious fervor.[4] These include

- *Scenario-based development.* We initially spent a great deal of time designing overly complex sensory systems for the room, without much thought regarding how we would eventually use them—not a good idea. In the end, when we started thinking up room demos, that is, scenarios to show off our work, what we wanted to do our sensors didn't support, and what our sensors supported, we didn't want to do. In particular, the sensing needs our desired demos required were actually simpler, albeit different, from what the complex computer-vision systems we had created provided. Much time could have been saved had we thought ahead.
- *Do not scavenge.* There is an enormous temptation when designing an intelligent room to try and incorporate all your friends' thesis work into it. How can you resist adding the latest state-of-the-art system that does [your favorite idea]? However, there is a good chance systems not intended to work together will refuse to do so smoothly and almost surely will be unable to take advantage of each other's capabilities. You must consider how components will integrate into the overall system when designing the individual components.
- *Systems should leverage off each other.* Because nothing is perfect, particularly in AI, throw everything you can at a problem. In the Intelligent Room today, computer-vision systems communicate with speech-recognition systems. A strange mix, you might think, but we have found that as you approach something, such as a projected map, you are more likely to talk about it. Thus, by visually locating a person in the room, we can cue the speech-recognition system with information about what the user is likely to say. In this way, we get higher speech-recognition accuracy. This principle generalizes, and many subsystems in the intelligent room intercommunicate with one another for similar reasons.

Big Brother, post-1984?

It is quite easy to trace almost all work in the intelligent environments back to the very influential Digital Desk project at Xerox PARC in the late 80s and early 90s, which was among the first user interfaces to directly observe people manipulating real objects. Using a video camera, it watched people reading real paper documents on the surface of a real desk—highlight something with your finger, and the system proceeds to scan in the delineated text. Surprisingly, however, the very first intelligent environment was proposed in the late 18th century by well-known British philosopher and would-be prison warden, Jeremy Bentham. Bentham designed the Pantopticon, an Orwellian structure in which a hive of rooms (or cells) could be kept under the constant scrutiny of an unseen Observer.[5] Denizens of the Panopticon, which Bentham proposed could be either inmates, the insane, or students, would never be precisely sure when they were being observed by the central Observer, namely the warden, doctor, or graduate advisor. Order would be maintained exactly because observed infractions might be harshly punished at some unknown later date.

As pointed out by Bentham, and subsequently elaborated upon by Michel Foucault, the observed confer enormous power on the Observer, and this might seem the most serious objection to widespread introduction of intelligent environments.[6] The potential for abuse is frightening and should certainly give any technology enthusiast or futurist pause. However, this need not be a fatal objection, and I think it is premature to address at present. Given that we have no clear idea what types of new sensing technologies will be developed and deployed in the future, worrying about security now is therefore somewhat pointless. Whatever privacy-guaranteeing techniques are developed for today's technology will almost surely be irrelevant for tomorrow's. More importantly, fear of misuse is no reason not to push for something that has the potential to so greatly revolutionize our lives.

In the end, it will come as no great surprise if the widespread acceptance of intelligent rooms, homes, cars, and so forth comes as much from clever marketing as from clever security. If someone were to propose filling your home with microphones that anyone in the world could listen to anytime, day or night, you might very likely shudder in horror. Yet, your home is filled with microphones—every telephone has one. But perhaps you object, "Phones can't be abused like that! Other people can hear me only when I let them!" Aha! It sounds as if you've already been socialized to accept telephones; it is quite difficult to now view them as a realistic threat. I think it is quite reasonable to expect that someday your children will be similarly comfortable inside their intelligent homes.

References

1. M. Coen, ed., *Proc. 1998 AAAI Spring Symp. Intelligent Environments*, AAAI TR SS-98-02, AAAI Press, Menlo Park, Calif., 1998.
2. M. Coen, "Building Brains for Rooms: Designing Distributed Software Agents," *Proc. Ninth Innovative Applications of AI Conf.*, AAAI Press, 1997, pp. 971–988.
3. B. Phillips, *Metaglue: A Programming Language for Multi-Agent Systems*, M. Eng. thesis, Massachusetts Inst. of Tech., Cambridge, Mass., 1999.
4. M. Coen, "Design Principles for Intelligent Environments," *Proc 15th Nat'l Conf. Artificial Intelligence*, AAAI Press, 1998, pp. 547–554.
5. J. Bentham, *The Panopticon Writings*, Verso, London, 1995.
6. M. Foucault, "The Eye of Power," *Power/Knowledge*, Pantheon Books, New York, 1981.

An intelligent environment must be adaptive

Michael C. Mozer, University of Colorado

What will the home of the future look like? One popular vision is that household devices—appliances, entertainment centers, phones, thermostats, lights—will be endowed with microprocessors that allow the devices to communicate with one another and with the home's inhabitants. The dishwasher can ask the water heater whether the water temperature is adequate; inhabitants can telephone home and remotely instruct the VCR to record a favorite show; the TV could select news stories of special interest to the inhabitant; the stereo might lower its volume when the phone rings; and the clothes dryer might make an announcement over an intercom system when it has completed its cycle.

The cost of the hardware infrastructure is not prohibitive if the devices are mass-produced and if communication is conducted over power lines or wireless channels. Even if the cost is too high today, wait a few years and the price will drop precipitously. Adopting a uniform communication protocol is also not an obstacle in principle. The real reason why this vision of home automation seems unlikely is that it requires a significant programming effort, and worse, the programming must be tailored to a particular home and family and must be updated as the family's lifestyle changes. Tackling the programming task is far beyond the capabilities and interest of typical home inhabitants. People are intimidated by the chore of programming simple devices such as VCRs and setback thermostats, never mind a much broader array of devices with far greater functionality.

Perhaps if you were Bill Gates, you might hire a full-time team of engineers to customize your system and keep it up to date. Some commercially available systems adopt this strategy on a smaller scale: following installation, a technician comes to the home, consults with the inhabitants, and sets up the initial programming. As the inhabitants' needs change over time, the technician can modify the programming, either remotely or on site.

The adaptive house

In contrast to existing automated homes that can be programmed to perform various functions, our research focuses on developing a home that essentially programs itself by observing the lifestyle and desires of the inhabitants and learning to anticipate their needs.[1–3] This house's intelligence lies in its ability to adapt its operation to accommodate the inhabitants; thus, we call the project the adaptive house.

Traditional automated homes require a user interface, such as a touchscreen that displays the system state and controls, or a speech-recognition front end. However, even a well-designed interface impedes the acceptance of an automated home. In contrast, the adaptive house should be unobtrusive and require no special interactions. Inhabitants operate the adaptive house as they would an ordinary home—using light switches, thermostats, and on/off and volume controls like those to which they are accustomed. Unlike an ordinary home, however, these adjustments are monitored and serve as training signals—indications to the house as to how it should behave.

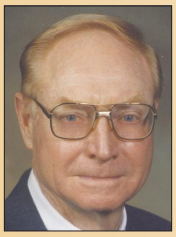
The adaptive house infers appropriate rules of operation of devices from the training signals and from sensors that provide information about the environmental state. As the house becomes better trained, it begins to anticipate the inhabitants' needs and sets devices accordingly, gradually freeing inhabitants from manual control of the environment. For example, it could automatically maintain the room temperature to a level appropriate given the particular occupants, activities, manner of dress, and time of year; it could choose one pattern of lighting while dinner is being prepared, and another for a late-night snack; it could turn on the television news during dinner, or play classical music when water is drawn for a bath, based on past selections of the inhabitants. Ideally, the house's operation is transparent to the inhabitants, other than the fact that they do not have to worry about managing the various devices in the home.

We might view the home as a type of intelligent agent that infers the inhabitants' desires from their actions and behavior. Intelligent software agents abound that attempt to satisfy the information needs of

Michael H. Coen is a doctoral candidate at the MIT Artificial Intelligence Lab, where he has lead the Intelligent Room Project for the past two years. His research focuses on intelligent rooms, software agent programming languages, multimodal sensor fusion, and coordination theories for multi-agent systems. He has SB and SM degrees in computer science from MIT. In 1998, he chaired the AAAI Spring Symposium on Intelligent Environments, the first such gathering of its kind. Contact him at the MIT AI Lab; 545 Technology Square; Cambridge, MA 02139; mhcoen@ai.mit.edu; www.ai.mit.edu/people/mhcoen.

Michael C. Mozer is an associate professor in the Department of Computer Science and the Institute of Cognitive Science at the University of Colorado. His research interests include computational modeling of human cognition, neural network architectures and learning algorithms for temporal pattern processing, and applications of machine learning to problems in engineering. He has a BA in computer science from Brown and an MA in psychology and PhD in cognitive science and psychology from the University of California at San Diego. Contact him at the Dept. of Computer Science, Eng. Center, Rm. 741, Univ. of Colorado, Boulder, CO 80309-0430; mozer@cs.colorado.edu; www.cs.colorado.edu/~mozer.

Richard Hasha is the director of engineering for Interactive Home Systems, a division of Corbis Corporation. IHS has overall responsibility for the specification, design, and implementation of all automated control systems for the Gates Estate at Medina, Washington. Richard was the chief systems architect for this project. He has over 27 years of broad-ranging software engineering experience, much of which has been focused on distributed computing. Contact him at IHS (Corbis), 15395 SE 30th Pl., Ste. 300, Bellevue, WA 98007; rhasha@hasha.seanet.com.



James L. Flanagan is the vice president for research at Rutgers University. His research interests include collaborative computing and multimodal human-computer interaction. He received a BS from Mississippi State University and an ScD and an SM from MIT, all in electrical engineering. He is a Life Fellow of the IEEE and a winner of the National Medal of Science, and a member of the National Academy of Sciences and the National Academy of Engineering. Contact him at the CAIP, Rutgers Univ., P.O. Box 1390, Piscataway, NJ 08855-1390; jlf@caip.rutgers.edu; www.caip.

Coming Next Issue

Idea Futures

by Robin Hanson

Robert Wood Johnson Foundation Health Policy Scholar, UC Berkeley

users of the Web. We extend the idea of intelligent agents to comfort needs of people in natural living environments. Intelligent agents seldom perform perfectly because of their limited ability to infer users' intentions. However, we can minimize this problem in natural environments through the use of smart sensors and some general domain knowledge, such as an analysis of typical tasks performed in an environment.

Residential comfort systems

To discuss the adaptive home in concrete terms, let us focus our discussion on the control of basic residential comfort systems: air heating, lighting, ventilation, and water heating. The reason for this focus is twofold:

- These devices are prime consumers of energy resources, and thus present an opportunity for energy conservation.
- Although some devices in the home are controlled with relative ease, such as the stereo or TV, the operation of residential comfort systems can be quite complex.

Consider the control problem for air heating. (In Colorado, we worry more about heating in the winter than cooling in the summer.) The thermostat could be set to 70° around the clock. However, this is inefficient because the house doesn't need to be heated while its inhabitants are at work, nor does the setpoint have to be as high at night. We could use a digital thermostat with multiple setback periods to specify when to lower the setpoint. However, different rules are required for weekdays and weekends. Furthermore, the setback thermostat only lets us specify first-order rules of occupancy (expected departure and return times based on weekday versus weekend). For efficiency, the thermostat should really know more subtle patterns of occupancy (expected return time based on day of week, departure time that day, weather conditions, and recent schedule, for example).

It must also consider the time required to heat the house, which depends on outdoor weather conditions. If the house has multiple furnaces or zoned control, room-occupancy patterns must be considered. Furthermore, alternative means of heating should be considered, such as opening blinds to allow for passive solar gains, electric space heaters to heat individual rooms, or fans to mix the air. Finally, utilities sometimes charge for energy based on time of use, making it more efficient to overheat the house during the day than to heat it to the appropriate setpoint immediately before the return of the inhabitants. Thus, regulating the air temperature in the house to simultaneously maintain comfort and energy efficiency is not a trivial challenge.

ACHE

We have constructed a prototype system in an actual residence. The residence was completely renovated in 1992, at which time the infrastructure needed for the adaptive-house project was incorporated into the building, including nearly five miles of low-voltage conductor for collecting sensor data and a power-line communication system for controlling lighting, fans, and electric outlets.

We call the system that runs the home ACHE, an acronym for *Adaptive Control of Home Environments*. At present, ACHE can control 22 banks of lights (each having 16 intensity levels), six ceiling fans, two electric space heaters, a water heater, and a gas furnace. ACHE has roughly 75 sensors, which include the following for each room in the home: intensity setting of the lights, status of fans, status of digital thermostat (which is both set by ACHE and can be adjusted by the inhabitant), ambient illumination, room temperature, sound level, status of one or more motion detectors (on or off), and the status of doors and windows (open or closed). In addition, the system receives global information such as the water heater temperature and outflow, outdoor temperature and insulation, energy use of each device, gas and electricity costs, time of day, and day of week. Figure 1 shows a floor plan of the residence, as well as the approximate location of selected sensors and actuators.

Objectives. ACHE has two objectives: anticipation of inhabitants' needs and energy conservation. For the first, lighting, air temperature, and ventilation should be maintained to the inhabitants' comfort; hot water should be available on demand. If inhabitants manually adjust an environmental setpoint, they are indicating that their needs have not been satisfied. For energy conservation, lights should be set to the minimum intensity required and hot water should be kept at the minimum temperature needed to satisfy the demand. Also, only rooms that are likely to be occupied in the near future should be heated; when several options exist to heat a room, the one minimizing expected energy consumption should be selected.

Achieving either of these objectives in isolation is fairly straightforward. If ACHE were concerned only with appeasing the inhabitants, the air temperature could be maintained at a comfortable 70° at all times. If ACHE were concerned only with energy conservation, all devices could be turned off. In what sort of framework can the needs of the inhabitants be balanced against energy conservation? We have adopted an *optimal control* framework, in which failing to satisfy each objective has an associated cost. A discomfort cost is incurred if ACHE does not anticipate inhabitant preferences. An energy cost is incurred based on the use of gas and electricity. ACHE's goal is to minimize the combined costs of discomfort and energy.

This framework requires that discomfort and energy costs be expressed in a common currency, which we have chosen to be dollars. Energy costs can readily be characterized in dollars, but some creativity is involved in measuring discomfort costs in dollars. Relative discomfort is indicated when the inhabitant manually adjusts a device (such as turning on a light); a misery-to-dollars conversion factor must be used to translate this relative discomfort to a dollar amount. One technique we have used to specify this factor relies on an economic analysis in which we determine the dollar cost in lost productivity that occurs when ACHE ignores the inhabitants' desires. Another technique adjusts the conver-

sion factor over a several-month period based on how much inhabitants are willing to pay for gas and electricity.

Prediction and control. To minimize combined discomfort and energy costs, ACHE must be able to predict inhabitant lifestyle patterns and preferences, and to model the physics of the environment. We illustrate with a simplified scenario: It's 6:00 pm and the home is unoccupied. ACHE must decide whether to run the furnace. On the one hand, if the furnace is turned on for the next half hour, ACHE predicts that the indoor air temperature will rise to a level tolerable by the inhabitants, but an energy cost will be incurred. On the other hand, if the furnace is left off and the inhabitants return home around 6:30, a discomfort cost will be incurred. The decision about the furnace state depends on the expected energy cost on the one hand and the expected discomfort cost on the other hand (which depends on the probability that the inhabitants will return).

ACHE thus requires predictors that estimate future states of the environment such as the probability of home occupancy (based on 30 variables, including recent occupancy patterns) and indoor air temperature (based on outdoor air temperature and a thermal model of the house and furnace), as well as inhabitant preferences such as the temperature required to keep the inhabitant from "punishing" ACHE. The predictors rely largely on neural networks, which are statistical pattern-recognition devices inspired by the workings of the brain. Neural networks can learn from experience—from data collected by the house.

We have conducted simulation studies of the air-heating system,[1] using actual occupancy data and outdoor temperature profiles, evaluating various control policies. ACHE robustly outperforms three alternative policies, showing a lower total (discomfort plus energy) cost across a range of values for the relative cost of inhabitant discomfort and the degree of nondeterminism in occupancy patterns.

We have also implemented and tested a lighting controller in the house.[2] To give the flavor of its operation, we describe a sample scenario of its behavior. The first time that the inhabitant enters a room (we'll refer to this as a trial), ACHE decides to leave the light off, based on the initialization assumption that the inhabitant has no preference with regard to light settings. If the inhabitant overrides this decision by turning on the light, ACHE immediately learns that leaving the light off will incur a higher cost (the discomfort cost) than turning on the light to some intensity (the energy cost). On the next trial, ACHE decides to turn on the light, but has no reason to believe that one intensity setting will be preferred over another. Consequently, the lowest intensity setting is selected. On any trial in which the inhabitant adjusts the light intensity upward, the decision chosen by ACHE will incur a discomfort cost, and on the following trial, a higher intensity will be selected.

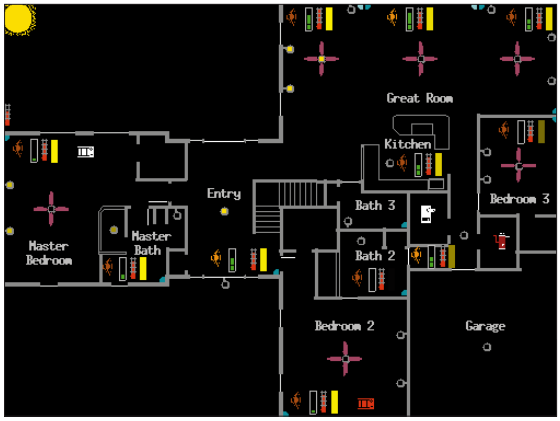


Figure 1. Floor plan of home equipped with the Adaptive Control of Home Environments (ACHE) system.

Training thus requires just three or four trials, and explores the space of decisions to find the lowest acceptable intensity. ACHE also attempts to conserve energy by occasionally testing the inhabitant, selecting an intensity setting lower than the setting believed to be optimal. If the inhabitant does not complain, the cost of the decision is updated to reflect this fact, and eventually the lower setting will be evaluated as optimal. As described in this scenario, ACHE relies on reinforcement-learning techniques.[4] ACHE includes a neural network that predicts when a room is about to become occupied, so that the lighting can be set prior to room entry. The scenario presented sidesteps a difficult issue: lighting preferences depend on the context (time of day, current activities, and ambient light level, for example), thus requiring ACHE to learn about desired lighting patterns in a context-dependent manner.

Discussion

Our research program hinges on a careful evaluation phase. In the long term, the primary empirical question we must answer is whether there are sufficiently robust statistical regularities in the inhabitants' behavior that ACHE can benefit from them. On first consideration, most people conclude that their daily schedules are not "regular"; they sometimes come home at 5 pm, sometimes at 6 pm, sometimes not until 8 pm. However, even subtle, higher-order statistical patterns in behavior—such as the fact that if you're not home at 3 am, you're unlikely to be home at 4 am—are useful to ACHE. These are patterns that people are not likely to consider when they discuss the irregularities of their daily lives. These patterns are unquestionably present, and our experiments to date suggest that they can be exploited to serve as the foundation of an intelligent, adaptive environment.

Acknowledgments

We are grateful to Marc Anderson and Robert Dodier, who helped develop the software infrastructure for the adaptive house. This research has been supported by the Sensory Home Automation Research Project (SHARP) of Sensory Inc., as well as a CRCW grant-in-aid from the University of Colorado, McDonnell-Pew award 97-18, and NSF awards IRI-9058450 and IBN-9873492.

References

1. M.C. Mozer, L. Vidmar, and R.H. Dodier, "The Neurothermostat: Adaptive Control of Residential Heating Systems," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, eds., MIT Press, Cambridge, Mass., 1997, pp. 953–959.
2. M.C. Mozer and D. Miller, "Parsing the Stream of Time: The Value of Event-Based Segmentation in a Complex, Real-World Control Problem," *in Adaptive Processing of Temporal Sequences and Data Structures*, C.L. Giles and M. Gori, eds., Springer Verlag, Berlin, 1998, pp. 370–388.
3. M.C. Mozer, "The Neural Network House: An Environment that Adapts to Its Inhabitants," *Proc. AAAI Spring Symp. Intelligent Environments*, M. Coen ed., AAAI Press, Menlo Park, Calif., 1998, pp. 110–114.
4. R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

Needed: A common distributed-object platform

Richard Hasha, Interactive Home Systems

Last year, a colleague and I decided we needed to pull our heads out of a very large home-automation project we'd been buried in and go see what the rest of the intelligent environments world had been up to. We attended the AAAI Intelligent Environments Symposium at Stanford in the spring of 1998. By the end of the conference, I understood that a great deal of energy is going into building supporting infrastructure just to get to a point where the interesting work can actually begin. I saw all these really smart people spending lots of energy on the same kinds of plumbing-related work just so they could begin focusing on their areas of interest. What a big waste of gray matter!

For the past five and a half years, I've been implementing the control system for a very large, very complex private home. The system is based fundamentally on a distributed-object model. Back when we were trying to figure out just how to structure the project, I spent a lot of time nailing down the core constraints and issues:

- It was clearly a distributed problem—lots of hardware and points of user interaction spread all over the place.
- You could not ask for a better example where an object-oriented design approach would be beneficial. This was clearly a modeling effort.
- The owners would want to continually incorporate interesting new functionality, as it became available.
- This was going to be a large system consisting of thousands of object instances scattered across lots of computers with tens of thousands of interobject relationships.
- We could not find a real-world example of such a system having ever been successfully built.
- People were going to literally live with this system day in and day out, so it could not be a fragile Tinkertoy.
- New features should have as little negative impact on existing functionality as possible.
- The system's core aspects would be very hard to change once the house was occupied.

This all pointed toward the need for a general, well-thought-out software platform that would stand the test of time as more and more functionality was piled on it. These facts drove us to spend a high percentage of our development resources on the construction of such a platform.

Given our experience, I very much believe that if such a platform were available to researchers and developers working on intelligent-environment projects, it would greatly amplify their efforts. This essay argues for such a software platform and further discusses some of the issues common to complex distributed applications such as those aimed at intelligent-environment behavior.

The problem domain

Here, I assume that most intelligent-environment R&D work will eventually lead to some form of distributed implementation as the various components of this work move from the programmer's workbench into a complete system. Lots of computers are dedicated to very specific tasks. In addition, the supporting software object model tends to map problem-domain entities, both physical and abstract, very precisely to their real-world counterparts—if they don't, terminal confusion quickly occurs. So, we end up with a complex system of interconnected but independent active objects—a network of objects, if you will. This object network can become quite complex. In fact, you would like to let it become as naturally complex as required by the specific project. Without an appropriate object-network platform (which we call an object-network OS), the right degree of natural object-model complexity is impossible, thus hampering overall progress in intelligent-environment R&D.

A prototypical object-network OS

What kind of functionality would be useful within the intelligent-environment R&D problem space? What key functional aspects should our hypothetical object-network OS provide? Some of the areas of functionality proven to be interesting within our project are support for large numbers of object instances, interobject referencing, interobject communication, monitoring and debugging facilities, and configuration management.

Support for large numbers of objects. To allow detailed and accurate problem-domain modeling, you want to support a distributed set of object instances numbering into the hundreds or, as in our case, the thousands. Both the supporting development tools and the object-network OS itself should provide a very simple means of adding new classes (abstractions) to the problem-domain model as well as allowing for multiple implementations of these abstractions. Furthermore, any number of runtime instances of these implementations should be able to be instantiated and allowed to coexist within the runtime object working set.

Within our project, we maintained a registry of both the abstraction derivation hierarchy and, in another dimension off this classification tree, zero or more implementation descriptions. These implementation descriptions describe not only the actual kind of object class used to make up a corresponding runtime instance but also the class of object used to actually configure instances of these runtime objects. Basically, the programmer thinks of this implementation description as a software part description. Once registered, any number of new runtime instances of an implementation can be configured and instantiated within the runtime model.

Also, object instances should be able to live on their own in an object-network OS. They should not require some other application-domain component to host them, as with many of the commercially available distributed-object platforms.

Interobject referencing. To support the naturally complex interobject connection requirement, an object-network OS must provide a means of cataloging and locating active objects. Beyond this, it must also provide a way to express and maintain interobject reference semantics in some other way than through hard (real) object references. Consider a simple two-object cross-reference situation: which object must come alive first? We obviously have a catch-22 situation.

In a system with hundreds or thousands of interobject references, not all of the modeled objects will be up at the same time—things fail, they're contained in different machines that start up at different times, and so on. And there are other real-world needs that lead us toward an environment that supports and promotes the idea that it's natural for objects to come and go in any order at any time. This is clearly an inherent

aspect of an active object-programming model.

For example, there's a class of behavior we on this project have termed *local survivability*. Objects and object clusters should do the best they can to provide the highest degree of intended functionality, even if some objects are missing from the complete runtime model. There are also very practical reasons for wanting to start and stop object instances arbitrarily in an operational environment. To allow for these situations, an object-network OS needs to support an object directory service and soft object references.

Object directory service. You assume that all active objects within the context of an object-network OS are named uniquely. For our project, we also found it useful to have every active object instance carry a description of the kind of abstraction it implements. We might also want to allow for the attachment of arbitrary classification attributes to object instances, thus allowing orthogonal views of the active object working set. To this end, an object-network OS needs to support an active (instantiated) object directory that allows any object within the system to locate and dynamically bind to other objects (at least by name) within that same system.

Soft object references. The object-network OS must support formation and maintenance of abstract references from one object to another without regard to the actual availability of the referenced object. This feature is key in allowing objects to come, go, and rebind in any order and at any time. This facility should also provide referenced object-availability notifications to any referencing objects.

Interobject communication

With an object directory service and soft object reference facilities within an object-network OS, we can construct whatever logical object-network topology we require.

Direct-method invocation. We assume that, via the soft reference facilities, we can achieve real object-method invocation for those referenced objects that are active. This direct object-method invocation is the obvious way two objects communicate. In our experience, there are at least two other forms of useful interobject communication that an object-network OS should support. These are packaged in the form of events and properties.

Distributed events. Generally, events are the *what*, *when*, *where*, and *why* of something worth noting within the context of a problem domain. *What* specifies the exact type of event that occurred. *Where* is the unique identification of the object that generated the event. In our implementation, we found that hierarchical classification of event types was very useful.

The basic event model that we found workable was a broadcast-and-subscribe model. An event-client object informs our object-network OS of its desire to receive event notifications by listening for specific types of events. An event's producer simply broadcasts the event out onto the object-network, and all interested objects will receive a corresponding notification. This broadcast-and-subscribe event model supports a very simple, loosely coupled, many-to-many class of interobject communication. Furthermore, by hierarchically classifying event types, we achieve the notion of abstract event monitoring.

Network properties. The other broad class of useful interobject communication supports the one-to-many relationship. Within our hypothetical object-network OS, we'll call this the *support-network properties*. This class of communication support provides publish-and-subscribe facilities for the property client and the property server. The property-server object publishes a named property to the world. Property-client objects then subscribe to a property by providing the name of the property-server object and the name of the published property. When the value of a property is changed (typically by the publisher of the property), the new property value is distributed to all interested property-client objects. In this model, the publishing object is unaware of any and all client interests, because our object-network OS has the job of actually distributing the activity. (The interobject referencing implied by network-property support is assumed to be built using soft object referencing so that the attributes of the soft references will be extended to the network properties.)

The combination of one-to-one (direct-method invocation), many-to-many (distributed events), and one-to-many (network property distribution) communication facilities are all very useful when building complex distributed applications such as those within the intelligent-environment problem space. Notice that, in each case, binding between objects is symbolic, further supporting arbitrarily complex object-interconnection topologies.

Monitoring and debugging facilities

To illustrate the need for integral monitoring and debugging facilities, let me use our project as an example. At last count, our deployed experiment had just over 6,000 active object instances, with an average of five external object references per instance—about 30,000 interconnects—all of which are spread across over 120 computers. In reality, mapping objects to computers is not all that important—in fact, this configuration is viewed as just a big network of 6,000 embedded computational entities. The question is not if a problem will occur but when it will occur. Unless our object-network OS supports a set of monitoring and problem-isolation facilities, we would be hopelessly lost and confused. In our implementation, many mechanisms

support this kind of activity. The two most important have proved to be logging and general object-state access.

Logging. I can't argue strongly enough for our object-network OS to support a ubiquitous, low-impact logging facility. It's equally important that object implementations are fully instrumented with log output and that such instrumentation never be removed. Once a system is having a problem, it's too late to think about adding logging to the code. We have had many complicated and hard-to-reproduce problems—what I'll call *n*th-order feedback classes of problems such as oscillation. Without the ability to snoop around and watch log activity, problems would be nearly impossible to diagnose.

Using our project's implementation as an example, let me describe what I think are the key aspects of a suitable logging facility. First, it is very useful to classify log-record output hierarchically. Typically, there would be classes and subclasses for maintenance, error, warning, and trace kinds of output. From the programmer's perspective, the logging primitives should be right there as part of their programming model and runtime environment. We keep a store-and-forward log facility object on each machine, whose job is to very quickly buffer local log activity into a disk-based FIFO queue, then forward that queue's content asynchronously to another facility (typically a central store that can be monitored on the fly). In addition, we can turn on and off different levels of log output at the object-instance level. Typically, our system runs objects with all trace output turned off, but if we're tracking down a problem we can reach in while an object is instantiated and modify its output log filter. This approach has proven essential in our system. Without such logging facilities, a system even an order of magnitude smaller would be impossible to operate.

General object-state access. The ability to probe the general state of an object instance while it's instantiated is also very useful. If we think about the general nature of the objects within our object-network OS, we see a high degree of common behavior. For example, any object can publish and subscribe to properties, listen for events, or reference and be referenced by other objects. In our implementation, this list of common traits is even longer. It would be very nice to be able to ask any object in the system to dump its current state. As with logging, this kind of facility is a necessity—for example, to monitor the system and verify that all of the referenced objects are up. This is a key ingredient in building a self-diagnosing system.

Configuration management

Some day, the components that make up the physical infrastructure of a facility (wires, building, room, city, and so forth) destined to behave as an intelligent environment might be completely self-identifying and, therefore, support self-configuration. But this utopian situation does not exist today. So, the question is, how do we deal with the configuration-management needs of an extensively componentized software environment such as the one we're discussing?

From our experience, you need to start with a single, core, configuration tool that can dynamically extend its behavior through implementation-aware configuration components. We call these *instance configurators*. The core configuration tool establishes a hierarchical user-interface environment and provides a common implementation component framework from which more application-specific instance configurators can be derived. The class-inheritance structure for the instance configurators parallels the corresponding runtime object-implementation classification hierarchy in almost every case. This leads to a great deal of reuse and overall enforcement of model-specific semantics within a general configuration framework.

Conclusions

We have demonstrated the need for an object-network OS that supports modular construction, configuration, and deployment of naturally complex software object models aimed at the intelligent-environment R&D problem space. This, in my view, would enable researchers to focus more on investigating intelligent environments and less on developing the infrastructure to support those investigations.

Through the deployment of appropriately complex and accurately modeled systems, the goal of intelligent environments will be realized.

Autodirective sound capture: towards smarter conference rooms

James L. Flanagan, CAIP Center, Rutgers University

Effective teleconferencing is increasingly sought for collaboration among groups who are geographically separated. Whether for activities as disparate as corporate strategizing, product design, or distance education, a key objective is voice communication that approaches the convenience and naturalness of face-to-face conversation—as though all participants were in the same meeting room. This desideratum implies "hands-free" sound pickup, where talkers may speak at some distance from the microphone system, without the encumbrance of hand-held, body-worn, or tethered equipment. Implied, too, is sound capture of sufficiently high quality—in which deleterious effects of multipath distortion (room reverberation) and interfering acoustic noise have been mitigated.

Enabling technology

We want, then, performance comparable to that of a microphone positioned close to each talker's mouth—even when the talker might be in motion (as in lecturing or explicating projected graphics). Several advances coalesce to support new capabilities for achieving this objective.

Recent years have seen development and large-scale manufacture of electret transducers. This device is a condenser microphone, in which the polymer membrane exposed to the sound wave has one surface metalized and the dielectric has been given a permanent electrostatic polarization (a bound charge that provides the bias potential for the condenser element).[1] Construction is exquisitely simple and inexpensive, and performance approaches that of the classic air-dielectric condenser microphone (flat amplitude response over a wide frequency range, linear phase, large dynamic amplitude, good sensitivity of the order of –40 dBv/Pascal, or about a millivolt of output for a strong conversational-level signal at 1 meter). In addition, no high-potential external bias is required, and the transducer has relatively low source impedance. In manufacture, a field-effect transistor preamplifier is typically integrated onto the condenser backplate, with the whole unit, in volume, costing less than a dollar.

Concomitantly, sampled-data theory,

digital-signal processing, and microelectronic circuitry have emerged explosively, so that sophisticated single-chip computers can be economically employed in great numbers. These advances, together with new acoustic understanding in characterizing the behavior of microphone arrays in enclosures (where geometries and acoustic properties are known, or prescribed) permit new approaches for sound pickup in large conference groups—approaches that go substantially beyond the shared microphone on the table or the tethered mike passed around from hand to hand.

Two steps are involved in achieving improvement: spatial location of the desired sound source (usually speech), and transduction of a good-quality replica of the desired sound. Both steps should be automatic.

Sound-source location. One approach to locating the position of a sound source in an enclosure (typically a talker) is based on acoustic measurements. Similar to electromagnetic techniques (Loran or GPS, among others) differences in time of flight to receivers of known position provide the relevant data. One implemented system uses two quads of microphones, placed preferably on adjacent orthogonal walls.[2] Each quad provides six distinct pairs of sensors for which differences in signal arrival time can be estimated. (That is, each set of four microphones (n), taken two at a time without regard to order (p), or $n!/p!\,(n-p)! = 6$.) Each time difference defines a hyperboloid-like surface on which a radiating source would produce the arrival difference. Intersection of the surfaces define a unique (overdetermined) point in 3D space.

The crucial measurement is that of time difference of arrival (TDOA). Cross correlation of the signals for each sensor pair helps to reveal the time-difference information. Computational advantages accrue from using fast Fourier transforms to accomplish the correlation in the frequency domain, namely by computing the normalized cross-power spectrum for each pair, inverse transforming, and judging the time shift that produces a maximum in the correlation function.[3,4] Also for computational efficiency, the source position's x,y,z coordinates are estimated by a nondirected gradient descent (using knowledge of enclosure geometry and sensor positions) to minimize the square error between the set of 12 measured TDOAs and those computed for any candidate x,y,z position.

In a perfect environment (or free space), the error can approach zero, but in a noisy, reverberant enclosure the TDOAs are contaminated. This impacts the practical choice of spacing for sensors in the quad. The spacing needs to be far enough to usefully measure a time difference, but not so far that the signal at each receiver is dominated by chaotic multipath reflections and interfering noise. In many conference rooms, a practically useful spacing is on the order of 20 cm. Figure 2 shows typical source-location accuracies measured by this technique in an industrial conference auditorium. As the measurements show, accuracies on the order of one-third meter are obtained in this practical environment.

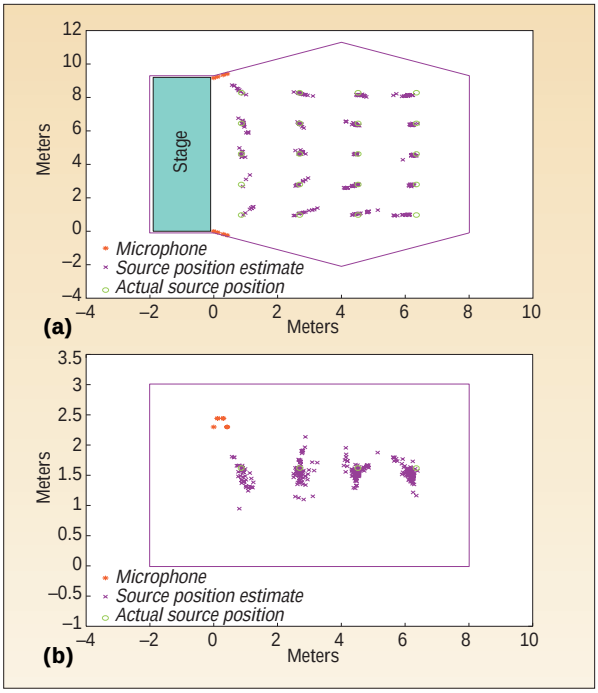


Figure 2. Accuracy of position estimates for acoustic (speech) sources in an industrial conference room: (a) top view, (b) side view.

Because most conference facilities include video equipment, source location by visual means is also an option, and a variety of techniques have been researched on this approach.[5,6] A combination of acoustic and visual methods is also an alternative and a current topic of research.

Sound capture. Having determined the location of the desired source, the next step is to capture a good-quality facsimile. One of the simplest means for mitigating reverberation and noise is delay-sum beamforming, where a cigar-shaped beam of sensitivity points at the source. A simple arrangement is the 1D line array in which delay is supplied to each receiver so as to cohere signals arriving from a prescribed direction.[7] For that direction (angle to the line), outputs of the multiple receivers add voltagewise, while for other directions, the uncorrelated components add powerwise, providing an array gain ideally bounded by the number of receivers. This works well if the environment is reasonably benign. But it has selectivity only in two spatial dimensions, and its signal-to-noise ratio (SNR) degrades monotonically as reverberation increases. (This is because the beamformer collects all sources along the bore of the beam—including those images of the source that it "sees" on any reflecting walls intersecting the beam.) What we need is spatial selectivity in range as well, giving full 3D selectivity. The technique of matched-filter processing applied to arrays provides this.

The matched filter requires convolution of each receiver signal by a causal approximation to the time reverse of the impulse response of the propagation path from the desired source to that receiver. Because the multipath characteristics are included, the matched-filter array can turn reverberant energy into useful signal, in effect making array performance (ideally) immune to reverberation and providing spatial selectivity in three dimensions. The computational costs are high, but digital-signal processing continues to enjoy a revolution of economy. In particular, if the impulse response from the desired source (focal) location to the nth sensor of the array is $h_{nf}(t)$,then the array output $O(t)$ for source signal $s(t)$ is given by the convolution

$$O(t) = \sum_{n=1}^{N} s(t) * h_{nf}(t) * h_{nf}(-t)$$

$$= s(t) * \sum_{n=1}^{N} \varphi(h_{nf}, h_{nf})$$

where $\varphi(h_{nf}, h_{nf})$ is the autocorrelation of the impulse response. On the other hand, a source at any location other than the focus produces a path impulse response $h_{nx}(t)$ and an array output

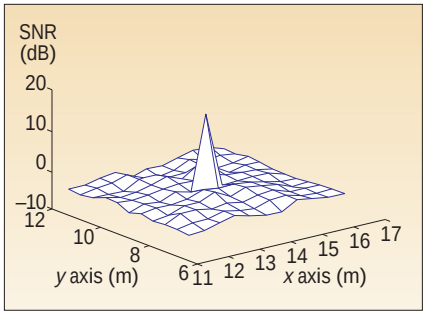


Figure 3. Computed signal-to-noise ratio for a 4-kHz bandwidth speech source matched-filter process by 100 sensors randomly distributed on one wall of a reverberant room of dimensions $20 \times 16 \times 5$ m. The acoustic absorption coefficient is $\alpha = 0.1$, and the array is focused at coordinates 14, 9.5, 1.35 m.

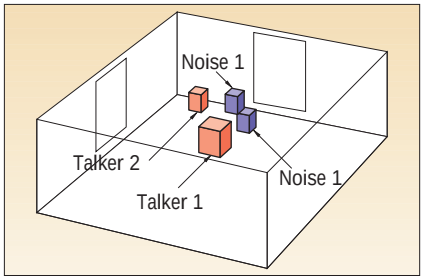


Figure 4. Multiple competing sources in the simulated room of Figure 3: Talker 1 man, Talker 2 woman, and Noise 1 and Noise 2 are Gaussian sources.

$$O(t) = s(t) * \sum_{n=1}^{N} \varphi(h_{nf}, h_{nx}),$$

where the cross correlation of the impulse responses $\varphi(h_{nf},h_{nx})$ (or rather the decorrelation of these impulse responses as the source moves away from the focal position) determines the spatial acuity in three dimensions.

The Fourier transform for $h_{nf}(t)$ is $H_{nf}(jw)$, where w is angular frequency, and each matched sensor contributes an output filtered by $|H_{nf}(jw)|^2$ for an on-focus source. For severe multipath this response is typically quite variable with frequency. But, where one sensor may have a spectral deficiency (zero), another may contribute fully. If the geometry of the array is chosen prudently, and N is somewhat large, $O(jw)$ can easily be rendered sensibly flat over the frequency range of interest.

In experiments, we found randomly distributed sensors located on orthogonal walls to be particularly advantageous.[8,9] Figure 3 illustrates computation of the behavior of a 100-sensor array, positioned on one wall of a moderately large room ($20 \times 16 \times 5$ m) having little sound absorption ($\alpha = 0.1$). For this figure, we measured the signal-to-reverberant-noise ratio for a 4-kHz bandwidth speech source in this room. When the source is located at the focal position (14.0, 9.5, 1.35 m), the improvement in SNR approaches 15 dB (which is adequately reflective of the ideal array gain of $10\log_{10}N$). To keep signal delay tolerable through the array processing for this simulated enclosure, all impulse responses have been truncated to 105 ms. (Two benefits accrue to the extent that the impulse responses can be truncated. Low-level "precursor" signals are diminished in the array output, and the filter computation is more economical.[10])

The 3D spatial selectivity of the matched-filter array is remarkably valuable in capturing a desired signal not only in the presence of reverberation, but also under adverse conditions of noise interference and competing signals. Two 2D random arrays in the room previously mentioned were applied to retrieving a designated speech signal in the presence of a simultaneous talker and two random noise sources, as Figure 4 shows. Talker 1 was a man and Talker 2 was a woman, and each speech source was 4 kHz in bandwidth. All source levels were of comparable acoustic power, except Noise 1 was –5 dB down from the other three. When received by a single microphone on the wall, this simultaneous signal complex is a totally unintelligible jumble. When the matched-filter array is focused on a desired source (Talker 1 in this instance), it can extract an intelligible signal. Figure 5 shows these comparisons along with the original source signal for Talker 1. The improvement in SNR afforded by the array over the single microphone is measured as 17 dB.

Preliminary system

As an initial exploration of automatic source location, coupled with slaved audio and video capture, we have implemented the experimental system of Figure 6. All computation is accomplished on a Pentium PC with one DSP board, and addressable bucket-brigade delay lines on each receiver accomplish the delay-sum beamforming (for a 21-element harmonically nested line array of first-order gradient electrets). The coordinates of the identified source are provided to both the slaved video camera and the beam-steered microphone array. Owing to limitations of arithmetic speed in the existing equipment, location estimates are made only at the rate of 2 sec^{-1}. Approximately twice this speed is desirable for tracking fast-moving talkers. Texas Instruments is sponsoring the design and construction of a fully digital high-speed system.

Research issues

Numerous research questions and applications have not been touched on in this brief discussion. A small sampling includes

- determination, storage, and truncation of coefficients for matched filtering, both from measurements in the room (with pseudo-random maximum-length sequences) and from on-the-fly computation (using room geometry and acoustical characteristics);
- techniques for zoom control of the focal volume and control of frequency response under zoom (a research project addressing these points is presently in progress under sponsorship of Intel Corp.);
- signal classification to determine the nature of sources (speech, music, noise);
- stability of filter coefficients to temperature, obstacles, room, and audience changes;
- automatic location of multiple moving talkers;
- sound projection in the receiving room to duplicate (virtual) source position; and
- sound reinforcement within large halls.

Going to large scale. Convention halls and large auditoria pose special problems for sound capture, reinforcement, and projection. Economical DSP and low-cost, high-quality electret microphones bode well for massive use of microphone arrays. Presently under study is one special hardware processor designed to control and operate up to 512 microphone channels.[11] Figure 7 illustrates a presently implemented 400-element array built for an earlier Bell Labs application. Sophisticated tracking of multiple talkers with matched-filter processing represents a sizeable technical challenge for arrays of this size.

Software conference manager. As we advance in capabilities for networked collaboration and large-group teleconferencing, opportunities increase for software agents and multimodal user interfaces. Combined with the technologies of speech recognition, speech synthesis, and talker identification, autodirective audio and video sys-

tems can alleviate the burdens on conference moderators.

Consider the scenario: Conferees convene for a meeting with a like group situated cross-country. As each arrives, he or she logs in, speaks an identification phrase, and is imaged by the face finder. Attendees also declare their credentials relevant to the conference's topic by speaking answers to brief questions from the synthetic voice at the enrollment station. As the conference unfolds, the slaved microphone array and the video camera track the active participants. The software manager (using keyword spotting, gisting analysis, talker identification, and the position data fed to the autodirective audio and video system) follows who's talking, for how long, and on what topic. If the "manager" observes that a participant of marginal credentials is talking too frequently on topics off the point, it ceases to point the microphone array and camera at that position!

Acknowledgments

The NSF, DARPA, NIST, Bellcore, IBM, TI, Intel, and the New Jersey Commission on Science and Technology have supported components of the research described here.

References

1. G.M. Sessler and J.E. West, "Self-Biased Condenser Microphone With High Capacitance," *J. Acoust. Soc. America*, Vol. 34, 1962, pp. 1787–1788.
2. D.V. Rabinkin et al., "A DSP Implementation of Source Location Using Microphone Arrays," *J. Acoust. Soc. America*, Vol. 99, No. 4, Pt. 2, Apr. 1996, p. 2503.
3. M. Omologo and P. Svaizer, "Acoustic-Event Localization Using a Cross Power Spectrum Phase-Based Technique," *Proc. ICASSP*, IEEE Press, Piscataway, N.J., 1994, pp. 273–276.
4. D. Rabinkin et al., "Estimation of Wavefront Arrival Delay Using the Cross-Power Spectrum Phase Technique," *J. Acoustical Soc. America*, Vol. 100, No. 4, Pt. 2, Oct. 1996, p. 2697.
5. Bellcore, "Introducing the Auto Auditorium System," 1998; http://AutoAuditorium.Bellcore.com.
6. J. Wang,Y. Liang, and J. Wilder, "Visual Information Assisted Microphone Array Processing In A High Noise Environment," *Proc. SPIE*, SPIE, Bellingham, Wash., Vol. 3521, 1998, pp. 198–203.
7. J L. Flanagan and E.E., Jan, "Sound Capture with Three-Dimensional Selectivity," *Acustica*, Vol. 83, No. 4, July/Aug. 1997, pp. 644–652.
8. D.V. Rabinkin, *Optimum Sensor Placement for Microphone Arrays*, PhD thesis, Dept. of Electrical and Computer Engineering, Rutgers Univ., New Brunswick, N.J., 1998.
9. E.E. Jan and J.L. Flanagan, *Image Characterization of Acoustic Multipath in Concave Enclosures*, Tech. Report TR-162, Rutgers Univ. CAIP Center, July 1993.
10. E.E. Jan, *Parallel Processing of Large Scale Microphone Arrays for Sound Capture with Spatial Volume Selectivity*, PhD thesis, Dept. of Electrical and Computer Engineering, Rutgers Univ., 1995.
11. H. Silverman et al., "A Digital Processing System for Source Location and Sound Capture by Large Microphone Arrays," *Proc. ICASSP 97*, IEEE Press, Vol. 1, 1997, pp. 251–254.

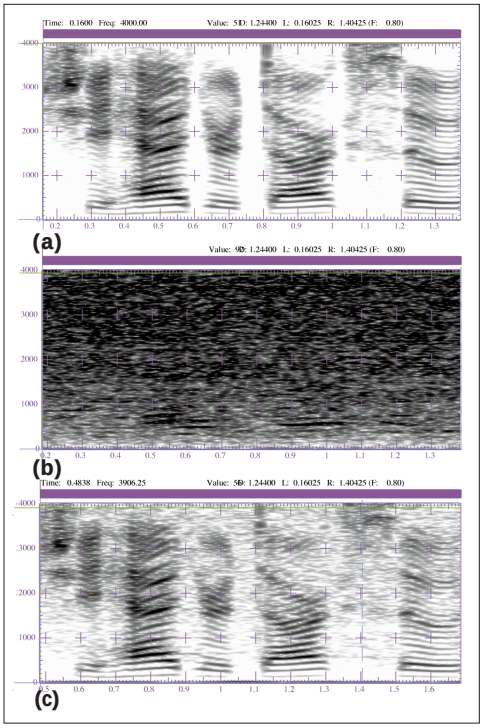


Figure 5. Retrieval of a desired signal from the simultaneous competing sources of Figure 4. Two matched-filter arrays of 100 randomly distributed sensors focus on Talker 1: (a) original source signal for Talker 1; (b) four simultaneous sources received by a single microphone; (c) Talker 1 retrieved by the matched-filter array from the simultaneous complex. (The abscissa is time in seconds; the ordinate is frequency in Hertz.)

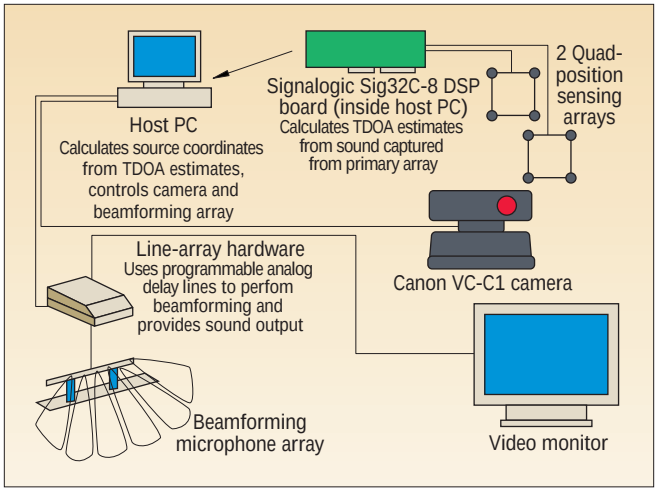


Figure 6. Implemented system for automatic source location and autodirective capture of audio and video.

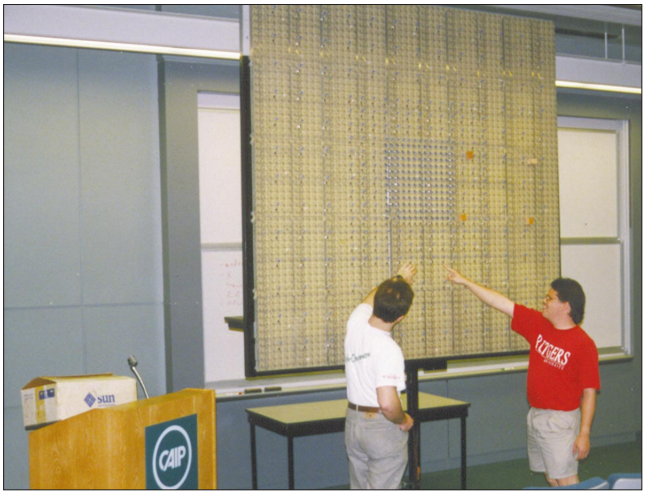


Figure 7. Planar array of 400 electret microphones for use in a large lecture hall.