Speech/Gesture Interface to a Visual-Computing Environment

Rajeev Sharma
Pennsylvania State University

Michael Zeller
University of Southern California

Vladimir I. Pavlovic, Thomas S. Huang, Zion Lo, Stephen Chu, Yunxin Zhao, James C. Phillips, and Klaus Schulten
University of Illinois, Urbana-Champaign

Although recent years have witnessed tremendous progress in 3D immersive display and virtual reality, the corresponding human-computer interaction (HCI) technologies have lagged behind. Current interfaces involve heavy headsets, datagloves, tethers, and other virtual reality devices, which may deter or distract the user. To fully exploit VR's potential for visualizing and interacting with complex information, users must be able to interact with the virtual display in a more natural way (see the "Hand gestures" sidebar on the next page).

In this article, we describe a bimodal speech/gesture interface, which we have integrated in a 3D visual-computing environment used by structural biologists. This interface lets researchers interact with 3D graphical objects in a virtual environment using spoken words and simple, free hand gestures. The reason we used a particular virtual environment context was to set the necessary constraints to make our analysis robust and to develop a command language that optimally combines speech and gesture inputs. Our interface uses

- automatic speech recognition (ASR), aided by a microphone, to recognize voice commands;
- two strategically positioned cameras to detect hand gestures; and
- automatic gesture recognition (AGR), a set of computer-vision techniques, to interpret those hand gestures.

The computer vision algorithms can extract the user's hand from the background, detect different finger positions, and distinguish meaningful gestures from unintentional hand movements.

Our main goal was to simplify model manipulation and rendering to make biomolecular modeling more playful. Researchers can explore variations of their model and concentrate on biomolecular aspects of their task without undue distraction by computational aspects. They can view simulations of molecular dynamics, play with different combinations of molecular structures, and better understand the molecules' important properties. A potential benefit, for example, might be reducing the time to discover new compounds for new drugs.

Virtual environment testbed

The Theoretical Biophysics Group at the University of Illinois, Urbana-Champaign, built the virtual environment we are using: MDScope. MDScope is a set of integrated software components that allows simulation and visualization of biomolecular systems in structural biology. As Figure 1 (next page) shows, three separate packages, which may be used individually or together, constitute the MDScope environment:[1]

We developed a speech/gesture interface that uses visual hand-gesture analysis and speech recognition to control a 3D display in VMD, a virtual environment for structural biology.

- The NAMD (Numerical Analysis Molecular Dynamics) program, a molecular-dynamics program that runs in parallel on various architectures and operating systems.
- The VMD (Visual Molecular Dynamics) program, a molecular-visualization program that displays both static molecular structures and dynamic molecular motion, as computed by programs such as NAMD.
- The MDComm (Molecular Dynamics Communication) software, which provides an efficient means of communication between VMD and NAMD and lets VMD act as a graphical user interface to NAMD. Using NAMD as a computational engine, VMD uses MDComm to provide an interface for interactive setup and display of a molecular-dynamics simulations on a remote supercomputer or high-performance workstation.

The NAMD program

Molecular-dynamics calculations are computationally very expensive and require large amounts of memory to store the molecular structure, coordinates, and atom-to-atom interaction lists. The challenge is to

0272-1716/00/$10.00 © 2000 IEEE

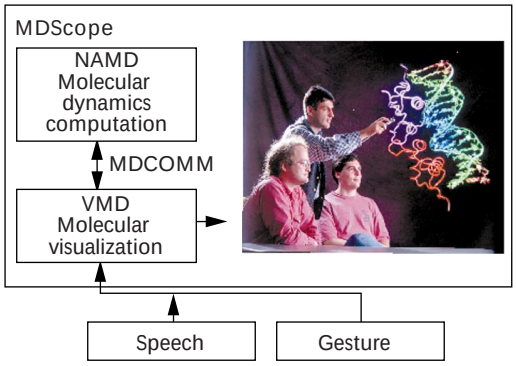
Hand Gestures

The communication mode that seems most relevant for manipulating physical objects is hand motion, also called hand gestures. People use this mode to act on the world, to grasp and explore objects, and to express ideas. Virtual objects, unlike physical objects, are under computer control. Psychological experiments indicate that people prefer to use speech and hand gestures in a virtual environment so that they can interact without special training or special apparatus and concentrate more on the virtual objects and the tasks at hand.[1] Several experimental systems study different aspects of this multimodality. One of the earliest was the "Put-That-There" system,[2] where a frame-based integration architecture fused spoken input and magnetically tracked 3D hand gestures. More recently, the QuickSet system integrated voice input with pen-based gestures to control military simulations.[3] Other systems include VisualMan,[4] Virtual-World, Artificial Life Interactive Video Environment,[5] and Smart Rooms.[6]

To interact naturally in a VR environment, users need as few devices attached to them as possible. However, most demonstrated gesture/speech systems use some hand-held device or instrumented glove, which is often tethered with cables that connect the device to a computer. This hinders user ease in interacting with the computer-controlled environment. A highly specialized application domain, such as simulation of surgery in a VR environment, might justify using such devices, but these cumbersome interface tools deter the everyday user. This problem has spawned active research toward video-based, noncontact gesture analysis that uses video cameras and computer vision to interpret gestures. Despite some progress in vision-based gesture recognition,[7] few systems integrate gestures in a working application.

References

1. A.G. Hauptmann and P. McAvinney, "Gesture with Speech for Graphics Manipulation," *Int'l J. Man-Machine Studies*, Vol. 38, No. 2, Feb. 1993, pp. 231-249.
2. R.A. Bolt, "Put-That-There: Voice and Gesture at the Graphics Interface," *ACM Computer Graphics*, Vol. 14, No. 3, 1980, pp. 262-270.
3. P.R. Cohen et al., "QuickSet: Multimodal Interaction for Distributed Applications," *Proc. Fifth ACM Int'l Multimedia Conf.*, ACM Press, New York, 1997, pp. 31-40.
4. J. Wang, "Integration of Eye-Gaze, Voice and Manual Response in Multimodal User Interface," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics*, IEEE Press, Piscataway, N.J., 1995, pp. 3938-3942.
5. P. Maes et al., "ALIVE: Artificial Life Interactive Video Environment," *Intercommunication*, Vol. 7, Winter 1999, pp. 48-49.
6. A. Pentland, "Smart Rooms," *Scientific American*, Apr. 1996, pp. 54-62.
7. V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, July 1997, pp. 677-695.



1 The VMD program, coupled with the NAMD program via MDComm. These facilities comprise MDScope, a problem-solving environment for structural biology. External speech/gesture user-interface components can control all aspects of VMD through a communications layer. On the right, three users discuss and manipulate a 3D structure of DNA.

efficiently calculate interatomic forces using high-performance computing. NAMD meets this challenge by using parallel computation and incorporating the Distributed Parallel Multipole Tree Algorithm (DPMTA). NAMD uses a *spatial-decomposition* algorithm to partition the task of computing the force on each atom among several processors. This algorithm subdivides the space occupied by the molecule into uniform cubes (or *patches*), as shown in Figure 2. The algorithm then distributes those patches among a parallel computer's processors. The processor to which each patch is assigned computes the motion of the atoms in that patch. As atoms move, they are transferred between patches, and patches are reassigned to different processors to maintain a uniform computational load.

The VMD program

The key functions of VMD are to visualize biomolecular systems, allow direct interaction between a user and a molecule being simulated on another computer, and provide an intuitive user interface for controlling the visual display and remote simulation. VMD provides various methods for rendering and coloring the structure, such as simple lines, solid bonds, and ribbon diagrams (see Figure 3). VMD uses the MDComm software to initiate, display, and control a simulation using NAMD. When NAMD calculates a molecular system's trajectory, it sends each atom's coordinates to VMD. Current net-

work technology provides the necessary bandwidth to communicate the atomic coordinate data; a high-performance dynamics program is crucial for furnishing new data at the speed required for interactive display.

VMD implements many different forms of user interfaces; users may control the program through keyboard commands, a mouse, or a graphical user interface. These controls let users modify the appearance of the molecules and display, control the display of the molecules' structural features, and access remote computers running molecular-dynamics simulations. Users can view multiple structures simultaneously and, because of a flexible atom-selection mechanism, easily select subsets of atoms for display.

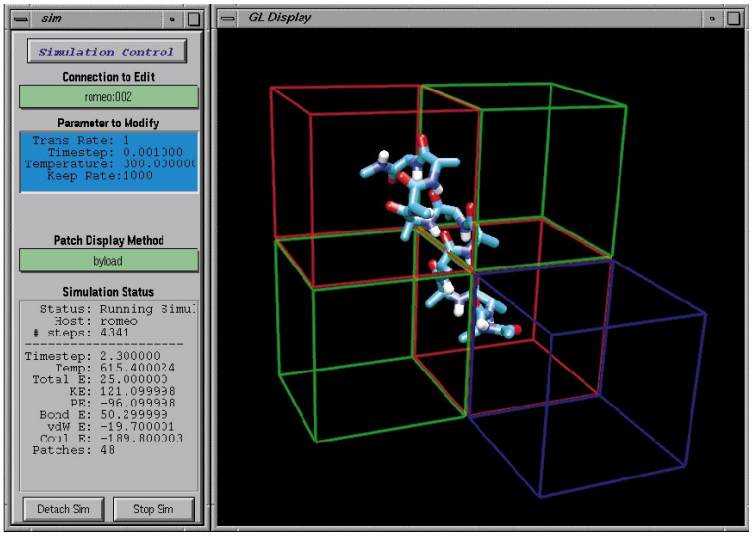
VMD includes an extensive text-command processing capability from the Tcl library, a popular and widely available package for script parsing and interpreting. Tcl lets users write scripts with features such as variable substitution, control loops, and function calls. Users can also save a molecule's current rendering in an image file or in a format suitable for use by several image-processing packages. By connecting directly to a remote computer running a molecular-dynamics simulation, VMD offers users the capability to interactively participate in an ongoing simulation—for example, the option to apply perturbational forces to individual atoms. VMD also implements a mechanism for external programs to serve as user-interface components by letting these programs communicate with VMD through standard network-communication channels. This makes possible new user-interface methods, such as the speech- and gesture-recognition systems discussed later, to be developed in parallel with VMD.

MDScope

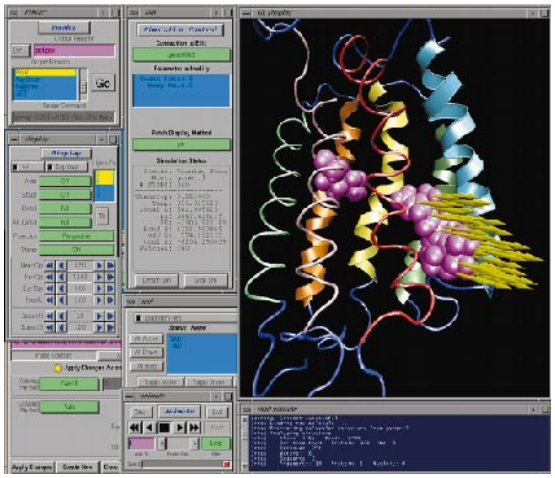
Development of MDScope is an ongoing project. Source code for MDScope applications are available via anonymous ftp at ftp.ks.uiuc.edu or on the Web at http://www.ks.uiuc.edu. This Web site also includes extensive documentation describing how to use and modify the programs. MDScope is available for various architectures, operating systems, and OpenGL-based workstations.

The primary infrastructure for our project is a large-screen stereographic projection facility, developed in the Theoretical Biophysics Group at the University of Illinois and shared by many biomedical researchers. The facility, designed for groups of up to 10 people, uses cost-effective, space-saving display hardware, which is added to a high-end graphics workstation and can be easily duplicated at other sites. It produces 8×6×6-foot 3D models in a 120-square-foot area. The facility includes a projector that displays alternating left- and right-eye views onto the screen at nearly twice the rate of ordinary projectors. The images, when viewed through special eyewear, produce a stereo display.

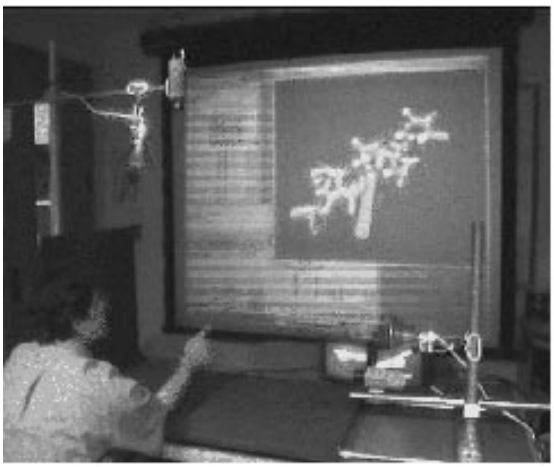
The primary VMD interfaces used by researchers are a keyboard and a magnetically tracked pointer. This is inconvenient because the system is typically used by multiple (six to eight) users, and the interface hinders the visualization system's interactive nature. Speech and hand gestures, on the other hand, are fundamental, natural methods of human communication; their use for interaction with and control of the display of VMD would greatly improve the program's utility. Thus, incorporating voice-command control in MDScope would free users of keyboards so that they could interact with the environment in a natural manner. The hand gestures would permit the users to easily manipulate the displayed model and play with different spatial combina-



2 VMD visualizing the spatial decomposition in NAMD. VMD displays a molecular-dynamics simulation of a small polypeptide computed by the program NAMD on a remote workstation and shown on a local graphics workstation. The form on the left gives information about the simulation state and controls for modifying the simulation parameters. The colored boxes surrounding the protein indicate NAMD's spatial decomposition. Each box's color indicates the relative amount of CPU time required to compute the motion of the atoms in that region (the red end of the color spectrum denotes a greater CPU time), thus providing a way of visualizing the distribution of computational load.



3 A VMD session demonstrating the graphical user interface and the text console. Molecules appear in the display window shown in the upper right corner; GUI components are on the left. Below the graphics library display window is the text console interface for VMD.



4 The experimental setup with two cameras used for gesture recognition.

(a) (b)

5 A sample pair of images from the two cameras used for automatic speech recognition: (a) top camera; (b) side camera.

tions of the molecular structures. Integrating speech and hand gestures as a multimodal interaction mechanism would be more powerful than using either mode alone, thus motivating the development of the speech/gesture interface.

Software

Our goal was to minimize the modifications needed to the existing VMD program for incorporating the new interface. The experimental prototypes that we built for both the speech and hand-gesture analysis required an addition to the VMD environment. To reduce the complexity and increase the flexibility of the program design, we added a communications layer so that external programs could be written and maintained independently of the VMD code. These programs use the VMD text language to query VMD for information or to send new commands. The VMD text language is based on the Tcl scripting language. Because all VMD capabilities are available at the script level, an external program can control VMD in any way. Both the ASR and AGR programs interact with VMD using this method. For a simple voice command, such as "rotate left 90," the ASR converts the phrase into the VMD text command "rotate y 90" and sends it to VMD. Similarly, when the AGR is used as a pointing device, it sends the commands to change the current position and vector of VMD's graphical 3D pointers.

Setup for visual gesture analysis

To facilitate the development of AGR algorithms, we designed an experimental platform (see Figure 4) for gesture-recognition experiments. Within the uniformly black background, a lighting arrangement shines red light on the hand without distracting the user from the main 3D display. The red light helps to localize the hand in video and track it robustly. An alternative would be to track the hand using skin-color segmentation or motion and shape information.[2] However, for the visualization setup, the ambient light is quite low, which makes tracking more difficult without the help of the additional lighting.

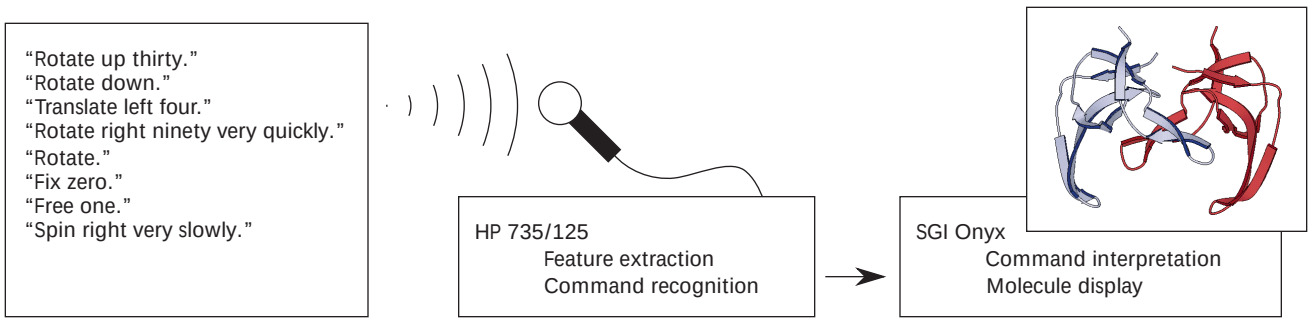
Figure 5 shows a sample pair of images from the two cameras. This setup lets a user sit at the table and use hand gestures to control the graphics display. Moreover, the setup can be transported easily and is relatively unobtrusive.

Setup for speech analysis

We have implemented a prototype ASR system and integrated it into VMD. The system has two blocks: the recorder front-end and the recognizer unit (see Figure 6). The recorder uses circularly buffered memory to implement its recording duties, sending its output to the recognizer unit in blocks. A digital volume meter accompanies this to give the user feedback by indicating an acceptable loudness range. We developed the recognizer using Hidden Markov Models. This unit performed feature extraction and time-synchronous Viterbi decoding on the input blocks, sending the decoded speech directly to the SGI Onyx workstation where the VMD process resides.

Speech/gesture command language

To effectively use the information the user inputs through spoken words and simple hand gestures, we designed a command language for MDScope that combines speech with gesture. This command language uses the basic syntax of `<action> <object> <modifier>`. The `<action>` component is spoken (for example,



6 Setup for the experiments on speech recognition for controlling the virtual display.

"rotate"); a combination of speech and gesture specify `<object>` and `<modifier>`. An example of this basic syntax would be speaking "this" while pointing, followed by a modifier to clarify what is being pointed to, such as "molecule," "helix," or "atom," and then speaking "done" after moving the hand according to the desired motion.

Another example of the desired speech/gesture capability is the voice command "engage," which would query VMD for the molecule nearest the pointer tip, make the molecule blink to indicate that it was selected, and save a reference to that molecule for future use. Once engaged, the voice command "rotate" would convert the gesture commands into rotations of the chosen molecule, and the command "translate" would convert them into translations. When finished, the command "release" would deselect the molecule and let the user manipulate another molecule.

Speech input using ASR

In integrating speech and gesture in the MDScope environment, we needed a real-time decoding of the user's commands to keep pace with the hand gestures. Thus, we needed *word spotting*, which means detecting a given vocabulary of words embedded in unconstrained continuous speech. Word spotting differs from conventional large-vocabulary continuous speech recognition systems, which seek to determine an optimal sequence of words from a prescribed vocabulary. A direct mapping between spoken utterances and the recognizer's vocabulary is implied with continuous speech recognition, leaving no room for the accommodation of nonvocabulary words in the form of extraneous speech or unintended background noise. Real-world applications dictate the basis for word spotting (also termed keyword spotting). Users of a spoken-language system often embellish their commands with supporting phrases and sometimes even issue conversation absent of valid commands. In response to such natural-language dialogue and the implications for robust human-computer interaction, we converted standard continuous-speech recognition systems into spotters by simply adding filler or garbage models to their vocabulary. The recognition output stream then contains a sequence of keywords and fillers constrained by a simple syntactical network. In other words, recognizers operated in a spotter mode.

Although early techniques emphasized a template-based dynamic time-warping slant, current approaches typically wield the statistical clout of hidden Markov models (HMMs), and recent ones come with the discriminatory abilities of neural networks. Typically word-based, they use an overall network that places the keyword models in parallel with the garbage models.

Keywords

Table 1 lists the keywords and their phonetic transcriptions chosen for our system. These commands let the VMD user manipulate the molecules and polymeric structures selected by hand gestures. In modeling the speech acoustics, we based the HMM system on phonemes rather than words, for large vocabulary flexibility in the given biophysical environment. Although implementing a word-based system would invariably be easier, retraining it would be inconvenient if the vocabulary changed.

Table 1. Keywords and phonetic descriptions for our system.

Keyword	Transcription
translate	t-r-ae-n-s-l-ey-t
rotate	r-ow-t-ey-t
engage	eh-n-g-ey-jh
release	r-ih-l-iy-s
pick	p-ih-k

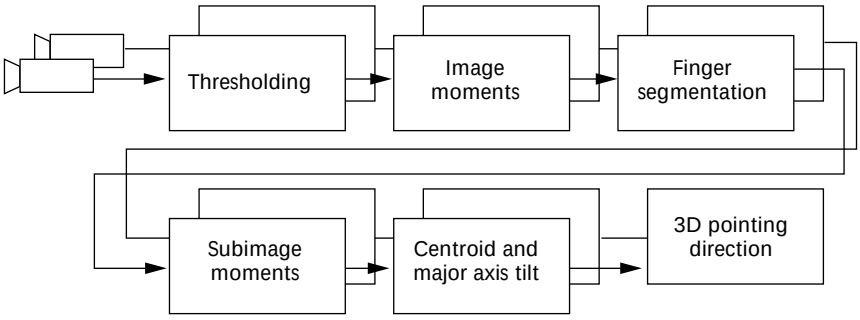
Table 2. Broad sound classes used as garbage models.

Sound Class	Symbol
Vowels-front	vf
Vowels-mid	vm
Vowels-back	vb
Dipthongs	dipth
Semivowels-liquids	svl
Semivowels-glides	svg
Consonants-nasals	cn
Consonants-stops-voiced	csv
Consonants-stops-unvoiced	csu
Consonants-fricatives-voiced	cfv
Consonants-fricatives-unvoiced	cfu
Consonants-whispers	cw

Fillers

Filler models are more varied. In large-vocabulary continuous-speech recognition applications, the non-keyword portion of the vocabulary can represent these fillers explicitly—for example, as whole words. In other tasks, non-keywords are built by a parallel combination of either keyword pieces or phonemes, whether they be context-independent monophones or context-dependent triphones or diphones.

We used 12 fillers (garbage models) to model extraneous speech in our experiment. Rather than using monophones or states of keyword models (as researchers have used in prior experiments), we used models that cover broad classes of basic sounds found in American English (listed in Table 2). Such models adequately cover the English language and are amenable to training. However, we modified these models in two ways. First, we did not use the class of "consonants-africates," because they don't occur frequently in the prescribed vocabulary or the training data. As observed by many researchers, varying or increasing the number of models does not gain much in spotting performance.[3] Second, we included a model for background silence in addition to the 12 garbage models listed. Such a model removes the need for an explicit endpoint detector by modeling the interword pauses in the incoming signal. Also, the descriptors for the vowel class correspond to the tongue hump's position in producing the vowel.



7 **An overview of the main steps involved in hand-gesture recognition.**

Recognition network

The recognition syntactical network placed the keywords in parallel to garbage models that included a model for silence. These models followed a null grammar, meaning every model may precede or succeed any other model.

Features and training

After sampling speech at 16 kHz and filtering to prevent aliasing, we preemphasized the speech samples with a first-order digital filter using a preemphasis factor of 0.97. We blocked these samples into frames of 25 ms with a shift between frames of 10 ms. We used a Hamming window to weight each speech frame; then we derived 16th-order mel-frequency cepstral coefficients and weighted them with a liftering factor of 22. We chose cepstral coefficients because they are more robust and discriminative than linear-predictive-coding coefficients or log-area-ratio coefficients. We also included normalized log-energy and first-order temporal-regression coefficients in the feature vector.

The HMMs' topology for both keyword phonemes and garbage models had five states, the three internal states being emitting states. Following a left-to-right traversal, each state was described by a mixture of five continuous-density Gaussians with diagonal covariance matrices. We used three iterations of the Baum-Welch reestimation procedure for training.

In training the 15 keywords, we developed 40 sentences. We individually paired each of the five keywords with the remaining four. We then doubled this pairing to provide a sufficient number of training tokens. Thus, the sentences contained keyword pairs such as "engage translate" and "rotate pick," which were arranged so that each keyword could be spoken 16 times. Each VMD user proceeded with this short recording session.

Training the garbage models required a far more extensive database of training sentences to provide an adequate amount of training data. The reason is that the 12 broad classes cover nearly the entire spectrum of the standard 48 phonemes. Subsequently, we used the Timit database to provide an initial set of bootstrapped models. We then conducted retraining for a VMD user who had recorded a set of 720 sentences of commonly used VMD commands. These sentences spanned the scope of the VMD diction, including a more detailed set of commands, numbers, and modifiers. This was necessary to normalize the data to the existing computational environment.

We trained the garbage models only once for this experiment. Hence, VMD users only needed to go through the short training procedure just explained.

Hand-gesture input using AGR

The general AGR problem is hard because it involves analyzing the human hand, which has a very high degree of freedom, and because the use of hand gestures is not well understood. However, we used the context of the particular virtual environment to develop an appropriate set of gestural commands. Our interface recognizes gestures by analyzing the sequence of images from a pair of cameras positioned so that they facilitate robust analysis of the hand images. The background is uniformly black to further help with the real-time analysis without using any specialized image-processing hardware.

Finger as a 3D pointer

The AGR system has two subsystem levels (see Figure 7). The first-level subsystem extracts a 2D pointing direction from single-camera images. The second-level subsystem combines the information obtained from the outputs of the first-level subsystems into a 3D pointing direction. To obtain the 2D pointing direction, the first-level subsystem performs a sequence of operations on the input image data. It first gives the gray-level image a threshold to extract a silhouette of the user's lower arm from the background. Next, the system calculates first- and second-image moments and uses them to form a bounding box to extract the index finger. Once the finger is segmented from the hand, another set of image moments is calculated, this time for the finger itself. Finally, based on these moments, the system determines the 2D finger centroid and finger direction. Finally, the system determines the 3D pointing direction of the hand, using the knowledge of the setup geometry and the 2D centroids. The system then forwards this information to the central-display manager, which displays a cursor at an appropriate screen position.

Gestures for manipulating 3D display

We also developed an AGR system based on HMMs to recognize basic manipulative hand gestures. We categorize these gesture commands as either dynamic (move back or move forward) or static (grab, release, stop, up, or down). The gestures used were mainly for manipulating and controlling the molecular structures' display (see Figure 8). We derived this gesture set by observing the MDScope system's functionality and the

corresponding command set used for interacting with it. In this way, we incorporated the gestures into the system without changing the existing command set.

The system uses image-geometry parameters as the features that describe any particular hand posture (static hand image). We use an image's *radon transform* to extract these features. The radon transform of the image $I(x, y)$ is defined as

$$R(\theta, t) = \int_{(x,y)} I(t\cos\theta - s\sin\theta,\ t\sin\theta + s\cos\theta)ds$$

where $0 \le \theta \le \pi/2$. The image-geometry moment of order k is then given by

$$m^{(k)}(\theta) = \int_t t^k R_0(\theta,\ t)dt$$

where R_0 denotes the radon transform, normalized with respect to the image mass:

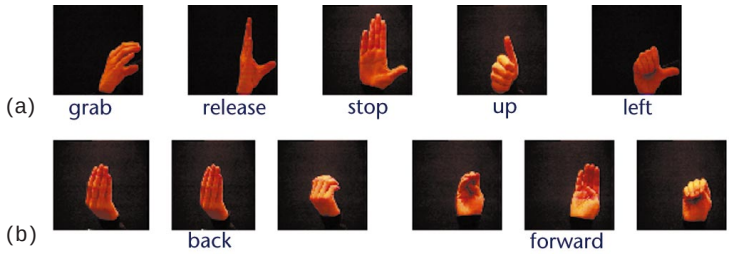
$$R_0(\theta, t) = \frac{R(\theta, t)}{\int_v R(\theta, v)dv}$$

The first-order moments constitute the image's center of mass. The higher-order moments provide additional information on image shape. We built the recognition system by training HMMs for the specific gestures on example runs. We modeled each gesture in the vocabulary as a single four-state HMM. We modeled the observations using a Gaussian mixture of two different sizes (one and three) with a diagonal covariance matrix.

Conclusions

Several researchers tested and evaluated the speech/gesture interface. Overall, new users needed only a few minutes to get acquainted with the setup. We asked the researchers to perform a generic task that involved selecting, moving, rotating, and releasing a certain part of a given molecular structure. This task would otherwise involve several keyboard commands or a combination of keyboard and magnetic-tracker input. Every user reported that working with the speech/gesture interface was more convenient and, in most cases, more efficient than the traditional interaction with VMD.

Upon testing the speech system as a whole with 50 test sentences that embedded the keywords within bodies of nonkeywords, we found a word-spotting accuracy of 98 percent on the trained speaker. The trained speaker refers to each user who trained the keywords, regardless of the one who trained the garbage models. The VMD users considered this very good for the given biophysics environment, thus supporting the techniques used. In general, false alarms occurred only for those situations where the user embedded a valid keyword within another word. For example, if someone says "translation" instead of "translate," the spotter still recognizes the command as "translate."



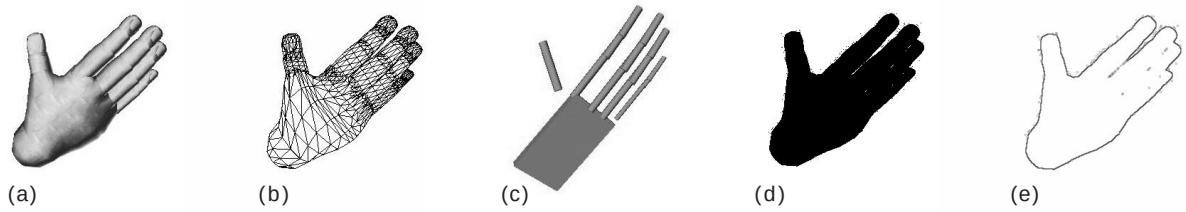
8 Examples of images from hand gestures used to manipulate a virtual object, and interpreted using AGR: (a) static gestures; (b) dynamic gestures.

Because of the strict grammar imposed on the allowed speech/gesture commands (which essentially makes the multimodal interaction sequential), the relative timing between the spoken words and hand gestures does not create problems in the interpretation.

The speech/gesture interface reported could be part of a more general multimodal framework, where other modalities such as gaze could also be exploited to make the interface more natural and efficient.

Using the simple setup with a uniform background, we segmented the hand image from the background in real time. This didn't require any restriction on the user other than being seated at the table, which is quite appropriate for the MDScope system. However, in a more general VR setting, we might need better segmentation techniques. The hand segmentation and corresponding motion analysis can benefit from the other modalities mentioned.

The experimental results for gesture recognition show that, even with simple image moments, the HMM-based approach yields a useful working system. However, a model-based approach can significantly affect recognition performance. For example, there is a trade-off between the reliability and speed of gesture recognition for different levels of the hand model used (see Figure 9).[2] One approach for AGR hand-motion analysis is to



9 Hand models of varying complexity: (a) 3D textured volumetric model; (b) 3D wire-frame volumetric model; (c) 3D skeletal model; (d) binary silhouette; (e) contour.

consider the motion class called articulated motion for analysis and tracking of the hand. Using the prediction based on articulated motion analysis, we can reliably derive a minimal description of the hand image in real time. The more detailed the hand mode, the better the prediction that can be made of the hand positions under different gestures. Such models can be the basis for developing a suitable feature vector for gesture classification and recognition.[4] The aim would be to replace the simple image moments in our current implementation with a feature vector that can define the more complicated hand gestures needed for manipulating a virtual environment.

Our focus on structural biology takes advantage of existing sophisticated software, provides concrete objectives, defines a well-posed task domain, and offers a well-developed vocabulary for spoken communication. Therefore, the VR setup we considered as a testbed for developing the multimodal interface facilitates the definition of gesture- and speech-recognition problems. Our prototype speech/gesture interface lets scientists easily and naturally explore the displayed information. Integrating speech and hand gestures as a multimodal interaction mechanism has proven to be more powerful than using either mode alone.

From a structural biologist's view, a robust gesture/speech interface for a molecular-graphics framework is useful for several reasons. First, it eliminates typing commands, which would require knowledge of the correct spelling and syntax. Pointing to a structure and saying "rotate this helix 75 left" is easier than entering the command using a keyboard, a mouse, menus, or a 3D tracker. Second, this novel interface simplifies accessing complex molecular-graphics programs for the novice or casual user. It also lets experienced users achieve many tasks in less time while focusing on the scientific merit of the modeling. Third, the gesture/speech interface is particularly useful for teaching and lecturing because it provides a far more natural way for presenting information than typing commands. Currently, because of the tedious task of controlling the VR display, a second person often accompanies the lecturer to operate the molecular-graphics program. Finally, the interface could help in the preparation of figures for publication. It can potentially provide a quicker way to explore different aspects of the model, select the most informative orientation of a biological structure, or simply add color and shading to the model to highlight specific features.

The combination of high-performance computing and high-end graphics for research in structural biology will open new avenues for very large-scale biomolecular modeling. A robust speech/gesture interface will offer a new level of interactive visualization not possible before. ■

Acknowledgments

We gratefully acknowledge the financial support of the National Science Foundation (grants IIS-97-33644, IRI-89-08255, BIR-9318159, IRI-95-02074, IRI-96-34618); the US Army Research Laboratory (Cooperative Agreement No. DAAL01-96-2-0003); Sumitomo Electric Industries; the National Institutes of Health (PHS 5 P41 RR05969-04); and the Roy J. Carver Charitable Trust.

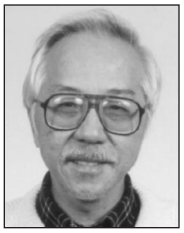
References

1. M. Nelson et al., "MDScope—A Visual Computing Environment for Structural Biology," *Computational Physics Communication*, Vol. 91, Nos. 1–3, Jan. 1995, pp. 111-134.
2. V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, July 1997, pp. 677-695.
3. J.G. Wilpon et al., "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models," *IEEE Trans. ASSP*, Vol. 38, No. 11, Nov. 1990, pp.1870-1878.
4. J.M. Rehg and T. Kanade, *Digiteyes: Vision-Based Human Hand Tracking,"* Tech. Report CMU-CS-93-220, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pa.,1993.

Rajeev Sharma *is an assistant professor in the Department of Computer Science and Engineering at Pennsylvania State University, University Park. He received a PhD in computer science from the University of Maryland, College Park.*

Michael Zeller *is Director of Systems Engineering at H&F Aeronautical Technologies, in San Diego. He received his Diplom degree and PhD in physics from the Johann Wolfgang Goethe University in Frankfurt, Germany.*

Vladimir Pavlovic *is a doctoral student in electrical engineering at the Department of Electrical and Computer Engineering at the University of Illinois, Urbana-Champaign.*



Thomas S. Huang *is the William L. Everett Distinguished Professor of Electrical and Computer Engineering at the University of Illinois, Urbana-Champaign, and a research professor at the Coordinated Science Laboratory. He also heads the Image Formation and Processing Group at the Beckman Institute for Advanced Science and Technology. He received his ScD in electrical engineering from the Massachusetts Institute of Technology.*

Zion Lo *received an MS in electrical and computer engineering from the University of Illinois, Urbana-Champaign.*

Stephen Chu *is a PhD candidate in the Electrical and Computer Engineering Department at the University of Illinois, Urbana-Champaign, where he also received an MS in electrical engineering.*

Yunxin Zhao *is an associate professor in the Department of Computer Engineering and Computer Science at the University of Missouri-Columbia. She received her PhD from the University of Washington, Seattle.*

James C. Phillips *is studying for his doctorate in physics at the University of Illinois, Urbana-Champaign, where he also received an MS in physics.*

Klaus J. Schulten *is a professor in the Physics Department at the University of Illinois, Urbana-Champaign. He received his Diplom degree in physics from the University of Munster, Germany, and his PhD in chemical physics from Harvard University.*

Readers may reach the authors in care of Klaus Schulten, Beckman Inst., 405 N. Matthews Ave., Univ. of Illinois, Urbana, IL 16801; kschulte@ks.uiuc.edu; or Rajeev Sharma, Dept. of Computer Science and Engineering, 220 Pond Lab, Pennsylvania State Univ., University Park, PA 16802; rsharma@cse.psu.edu.