Real-Time Animation Software for Customized Training to Use Motor Prosthetic Systems

Rahman Davoodi and Gerald E. Loeb, *Senior Member, IEEE*

***Abstract*—Research on control of human movement and development of tools for restoration and rehabilitation of movement after spinal cord injury and amputation can benefit greatly from software tools for creating precisely timed animation sequences of human movement. Despite their ability to create sophisticated animation and high quality rendering, existing animation software are not adapted for application to neural prostheses and rehabilitation of human movement. We have developed a software tool known as MSMS (MusculoSkeletal Modeling Software) that can be used to develop models of human or prosthetic limbs and the objects with which they interact and to animate their movement using motion data from a variety of offline and online sources. The motion data can be read from a motion file containing synthesized motion data or recordings from a motion capture system. Alternatively, motion data can be streamed online from a real-time motion capture system, a physics-based simulation program, or any program that can produce real-time motion data. Further, animation sequences of daily life activities can be constructed using the intuitive user interface of Microsoft's PowerPoint software. The latter allows expert and nonexpert users alike to assemble primitive movements into a complex motion sequence with precise timing by simply arranging the order of the slides and editing their properties in PowerPoint. The resulting motion sequence can be played back in an open-loop manner for demonstration and training or in closed-loop virtual reality environments where the timing and speed of animation depends on user inputs. These versatile animation utilities can be used in any application that requires precisely timed animations but they are particularly suited for research and rehabilitation of movement disorders. MSMS's modeling and animation tools are routinely used in a number of research laboratories around the country to study the control of movement and to develop and test neural prostheses for patients with paralysis or amputations.**

***Index Terms*—Modeling, motor neural prostheses, real-time animation, virtual rehabilitation.**

I. Introduction

HUMAN movement and its control are being studied in many laboratories and in many disciplines. When injuries or diseases cause movement disorders such as paralysis, amputation or stroke, normal movement must be restored by the use of prostheses or physical therapy and rehabilitation, often in combination. The sheer complexity of the neuromuscular system makes it inefficient to design and apply effective treatments by building and testing physical systems incrementally. Developers in many fields increasingly turn to mathematical models, computer simulations and virtual reality displays to estimate the functionality and usability of hypothetical designs.

Reanimating paralyzed limbs, for example, requires electronic devices that can precisely control the activation of paralyzed muscles to move the limb in a useful manner [1]–[3]. Similarly, prosthetic limbs with multiple degrees-of-freedom (DOF) must be controlled by voluntary commands of the amputee patient [4], [5]. Further, the patients must be trained in controlled and safe environments to generate the voluntary commands to control effectively the movement of their paralyzed or prosthetic limbs. Modeling and simulation in virtual reality environments is one of the techniques that have been proposed to optimize the design of neural prostheses, fit them to the patients, and train the patients to operate their prostheses effectively. Similar techniques can be used to treat patients with neurological disorders to regain their normal function and to study the neural control of movement. These applications, however, rely heavily on the availability of software tools that can help the users build sophisticated models of biological and prosthetic limbs and to animate them in interactive virtual reality environments that depict motor tasks.

Currently, because of their size, the movie and video game industries are the main driving forces behind the development of new computer animation software. The vast array of proprietary and commercial software tools (e.g., Maya from Autodesk Inc.) developed for these industries focus mainly on creating the illusion of reality using a variety of sophisticated techniques [6]–[8] that are mostly unsuitable for neural prostheses and rehabilitation applications. The main purpose of these software tools is to synthesize motion trajectories between key frames or process prerecorded motion data from real characters by techniques such as blending, retargeting, and synchronization to create realistic looking motions for the virtual characters. Computationally intensive rendering is done offline to produce the final high quality animations with realistic appearance but the models need not be physiologically realistic and the simulations do not provide insights into the kinetics of their movements and interactions. These tools are not designed to deal with the real-time requirements of neural prosthetic applications, and they don't interface with commonly used hardware for capturing the movement and neural command inputs from patients. Therefore, biomedical researchers and developers have used commercial software specifically developed for virtual rehabilitation or visualization libraries that have been customized to specific application by adding additional routines and models, or they had

Manuscript received March 14, 2011; revised September 14, 2011; accepted October 28, 2011. Date of publication December 16, 2011; date of current version March 16, 2012. This work was supported by The Defense Advanced Research Projects Agency under Contract N66001-10-C-2009. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

The authors are with the Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: davoodi@usc.edu; gloeb@usc.edu).

Digital Object Identifier 10.1109/TNSRE.2011.2178864

to spend a great deal of time and effort to develop their own computer animation tools.

A natural application for virtual environments has been the training of amputee patients to operate their myoelectric prostheses, which tends to be difficult, time consuming, and stressful to patients. For example, Eriksson *et al.* [9] used an animated hand to investigate the feasibility of electromyogram (EMG) control of a prosthetic hand with multiple DOF. The animated hand represented a prosthesis with the desired mechanical DOF that did not exist in reality. In a study by Sebelius *et al.* [10], the movement of a multi-fingered prosthetic hand that was not available at the time was represented by the animation of a human hand in a virtual environment. Such virtual simulations enabled the investigators to test the viability of novel prosthetic concepts and refine them before they were actually manufactured, thereby significantly reducing the costs associated with building physical prototypes that may turn out to be invalid. A similar approach was used by Soares *et al.* [11], in which the animated arm consisted of skinned meshes of the human arm with a realistic appearance and was used to visualize the postures of a prosthetic arm corresponding to the patient's EMG commands. Discrepancies between the virtual human arm and the prosthetic arm it depicts may hinder the developer's ability to test the viability of the prosthetic arm designs and may also reduce the effectiveness of the virtual training and its transfer to the real world. Further, the animations in this study were limited to rendering the selected static postures and therefore did not have to support continuous animations of the arm in real time. Dupont and Morin [12] developed a system to train and assess the myoelectric control of upper limb prostheses by child amputee patients. The training system used simple computer visualizations of the hand posture. Evaluation of the training system by fifteen nonamputee adult volunteers showed that all subjects improved at myoelectric control and that they improved more in the early stages of training than in later stages. The animations in this study were found to be effective in training the operation of the hand opening and closing but training to operate newer hands with multiple DOF requires more realistic animations that can mimic the appearance and behavior of the prosthesis and can respond to the users' commands in a timely manner. Nishikawa *et al.* [13], demonstrated the efficacy of real-time training in virtual environments where they used primitive animations of the prosthetic hand in place of the real prosthesis. But the trained motions were the desired postures the patients must achieve and did not include dynamic movements or interactions with objects in the task environment. In a training system developed by Kuttuva *et al.* [14] the pressure between the socket and the stump in lower arm amputees was used as the command signal to operate a virtual hand performing pick and place or pegboard exercises. The training tasks in this study were similar to their physical rehabilitation counterparts in clinical environments and therefore provided a more realistic environment for development and testing of next generation prosthetic limbs and prosthetic control systems. Virtual training of patients to operate their myoelectric prostheses has unique advantages, especially in early stages of recovery from amputation when the stump may not be able to bear the load of the real prosthesis. To optimize the outcomes, the training of the patients must start as soon after the amputation as possible [15], so it would be advantageous to begin training with a virtual arm whose appearance and behavior are as close as possible to the real prosthesis that will be fitted later.

Movement rehabilitation and training is another important area of application for virtual reality technologies. Piron *et al.* [16] used virtual rehabilitations, mimicking clinical rehabilitation tasks, to train stroke patients. The distinguishing feature of this study was the use of enhanced feedback such as the simultaneous plots of therapist's and patient's hand trajectories in 3-D space, which resulted in improved learning. These types of enhanced feedback are unique to virtual rehabilitation and could not be provided in conventional rehabilitation. A number of other studies have also shown that virtual training of stroke patients improves their function both in virtual and in real world tasks [17], [18]. Zhang *et al.* [19], [20] developed a virtual reality environment for telerehabilitation of stroke patients where the patient mimicked the desired movements made by primitive avatars. The avatars could animate simple reaching movements using data stored in motion files. The avatars had simple appearance, however, and could not be used to animate forearm and hand movements or interactions with the environment that are essential for the virtual task realism and patient motivation [21]. Further, the animation software was not general-purpose and therefore could not be easily adapted for other applications. Holden *et al.* [22] developed a virtual training system that used animations mimicking real rehabilitation tasks to help with motor training of stroke patients. Their motivation was to offer a home-based training system to patients who are increasingly getting less rehabilitation and are leaving the hospitals with lower functional levels. They clearly showed that the virtual training of stroke patients transferred to real world tasks. Virtual training is not limited to those with movement disability. Baek *et al.* [23] rescaled and retargeted the movement of an athletic trainer to match the body size of the trainee. The retargeted teacher movement was then animated along with the movements of the trainee's own movement. The superimposed animation provided immediate knowledge of results that allowed the trainee to learn to follow the teacher movement.

Another emerging area for the application of virtual environments in movement science is the study of motor control and learning in the central nervous system. In various studies (e.g., [24]–[26]), human or nonhuman primates have been placed in virtual environments where they could generate neural signals from different areas of their brain to drive an animated cursor (usually in the form of a sphere on a 2-D or 3-D computer display) to target positions. These test bed environments are helping researchers understand how the brain controls the movement of the hand in space and how these cortical neural signals may be used as a direct source of command to operate mechatronic prostheses for amputees and functional electrical stimulation to reanimate paralyzed limbs. As the field progresses, more sophisticated test beds including virtual models of the actual limb and rehabilitation tasks are becoming essential for further development [27]. Studies with nonhuman primates require extensive training with apparatus and tasks that scale gradually in complexity. Building multiple physical devices that can withstand abuse is time-consuming. Getting animals to generalize from their prior experience with physical objects to virtual en-

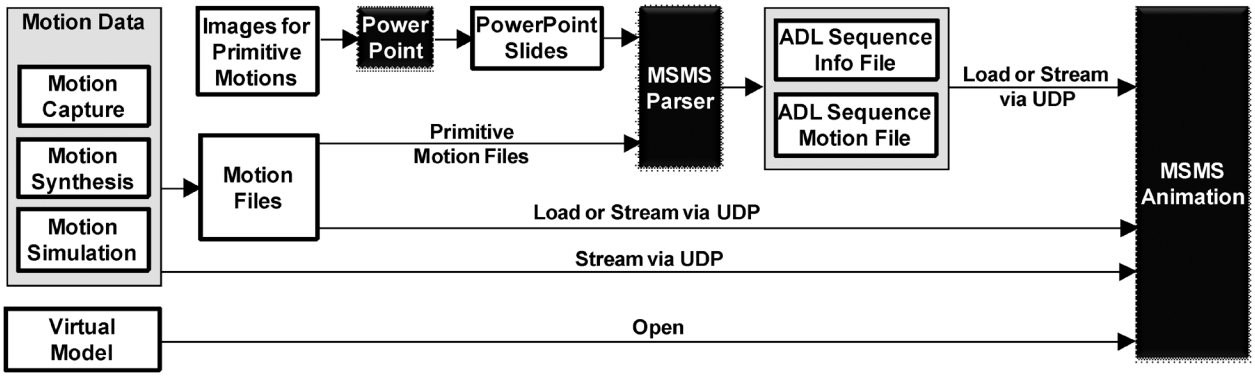


Fig. 1. Sources of motion data and types of animations in MSMS. The movement of a virtual model and physical characteristics of objects in MSMS can be animated by data from variety of offline and online sources. Complex and precisely timed animations of activities of daily living can be assembled from library of primitive movements in PowerPoint.

vironments depends on the verisimilitude of both the dynamics and the appearance of the simulation.

The studies above clearly demonstrate the need for a general-purpose software tool that can help researchers and developers in different areas of movement science to build models of limbs and tasks rapidly and to animate them in real-time for interactive virtual reality environments. We have developed a software tool known as MSMS that provides extensive modeling and animation capabilities specifically designed to facilitate rehabilitation and restoration of movement to patients with a variety of movement disorders. It allows building virtual reality environments where human or nonhuman subjects can learn to produce voluntary commands such as EMG or cortical neural activity to control the movement of a virtual limb and its interactions with the objects in the environment. In this paper, we are focusing on MSMS's unique animation capabilities and their potential to facilitate development and application of therapies by means of model systems and virtual environments. The reason for this focus is that most neural prosthesis and rehabilitation applications use animation-only simulations (see above). Nevertheless, MSMS has the tools to create realistic physics-based simulations of the limb and its interactions with external objects that are described elsewhere [28].

II. Building Blocks of MSMS Animations

Customized animations in MSMS are comprised of virtual models and motion data from offline or online sources (Fig. 1). The virtual models of a specific limb and the task environment can be built using interactive graphical tools in MSMS or imported from other software. Both the data for animation of the limb and objects and the physical appearance of objects such as their color and size can be read in from a file or streamed from live sources. The latter is essential for building interactive virtual reality applications where the subject interacts with the virtual model in real time.

A. Virtual Model

MSMS provides tools to build 3-D models of humans, animals, prosthetic limbs, and objects in the task environment. The segments and joints in these models must be designed to accommodate all the movements and interactions required by the specific application. The models can be assembled using MSMS

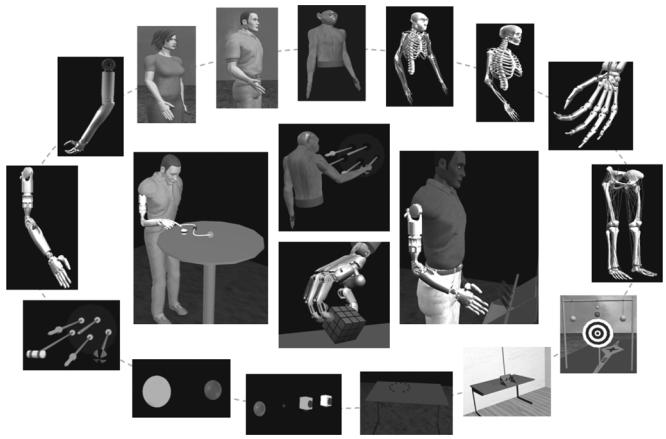


Fig. 2. MSMS's graphical tools can be used to interactively build sophisticated models of human, animal, and prosthetic limbs, and rehabilitation tasks. The user's library of limb and task models can be quickly combined in MSMS to build complete virtual representations of neural prostheses and rehabilitation tasks.

library of components or imported from other popular modeling environments. For example models of musculoskeletal systems and prosthetic limbs can be imported from OpenSim (SimTK.org) and SolidWorks (SolidWorks Corp.), respectively. Once a library of limb and task models are constructed, the user can quickly combine any limb with any task to build new virtual worlds. This feature is particularly useful for neural prostheses and rehabilitation applications where a patient may try a number of prosthetic limb designs to find the best match and may be required to train in different task environments to master the operation of the prosthesis. Fig. 2 shows sample virtual models constructed in MSMS and their combination to build virtual worlds. MSMS's support for user lights, user cameras, customized textures, sound playback, and 2-D and 3-D stereoscopic displays enable the design of virtual environments with the desired level of sophistication and realism.

The models built in MSMS are saved in standard XML (eXtensible Markup Language) file format. The use of XML format facilitates model parsing and validation using standard programming tools and the development of translators to import/export MSMS models to other formats used by other modeling software. More importantly, MSMS virtual models in their entirety

or individual parts (limb or task models) can be shared readily with other users.

B. Sources of Motion Data

Depending on the application, the motion data for animation can be obtained from one of three sources: motion capture systems, motion synthesis, and physics-based simulations. The motion data obtained from these sources can be stored in a motion file for later use or streamed directly to MSMS via user datagram protocol (UDP; see below) for real-time animations of the virtual model (Fig. 1).

Motion Capture Systems: Measuring the movement of a human subject is the preferred method for producing realistic animations. Both optical and electromagnetic motion capture systems can be used for this purpose. The recorded motion data is usually in the form of 3-D coordinates of the markers (in optical systems) or 6-DOF translations and rotations of the segments (in electromagnetic systems). These motion data must be transformed to the joint coordinates by deriving the angular position of each DOF.

Motion Synthesis: As an alternative to capturing the human movements, the motion data can be synthesized artificially. The synthesizer programs can use well known characteristics of human movement to create animation data. For example, smooth trajectories with bell-shaped tangential velocity curves can be computed between two postures. This approach is limited to producing stereotypical point to point movements and cannot be used for more complex or pathological movements that can only be captured from human subjects. MSMS facilitates the synthesis of the motion data between two postures by allowing the user to position the model in various postures (e.g., the initial and final postures of a point-to-point movement) and to export the corresponding motion files. The synthesizer program can then use these posture files as inputs and interpolate the desired number of data frames between them. The synthesized motion data can then be stored in MSMS motion file or streamed directly to MSMS via UDP.

Physics-Based Simulation of Movement: Another method for generating realistic motion data is physics-based simulation of the virtual models. The virtual models in MSMS can be automatically converted into a Simulink model that can be used to simulate the physics-based movement of the virtual model. When executed, the physics-based models in Simulink can predict the realistic movement of the virtual model in response to control inputs and external forces. The predicted motion data can be stored in motion files for later use or streamed to MSMS via UDP in real-time. The latter allows the user to view the simulated movement as an animation in real-time.

C. MSMS Motion File

MSMS uses an ASCII text file format to store the motion data for animation of the articulations in the virtual model. The MSMS motion file is comprised of a header row listing the names of the DOF in the model. In the header row, one has to list only the joints that must be animated and omit the others whose posture does not change. The header row is followed by any number of time-stamped data rows containing the motion data for the joints listed in the header row (Fig. 3). The data in each data row consists of the time and positions of the DOF listed in the header row at that time frame. For large models with many joints, building the header row is time consuming and prone to errors. MSMS has a utility that can automatically create a template motion file with the correct header row. The template motion file can then be edited manually in a text editor or spreadsheet or automatically by other programs.

Time	Sh_ABAD	Sh_IER	Sh_FE	Elbow_FE	Wrist_FE	Wrist_RUD
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.30	0.15	0.08	0.00
0.02	0.00	0.00	0.60	0.30	0.15	0.00
…….	…….	…….	…….	…….	…….	…….
2.00	0.00	0.00	60.00	30.00	15.00	0.00

Fig. 3. MSMS motion file format. Header row listing the names of the DOF for each joint is followed by the motion data in time-stamped rows.

D. MSMS Real-Time Animation Protocol

The motion data for animation of MSMS models can be sent using UDP protocol over a local network. This allows the animation data to be sent from a program running on the same PC or other networked PCs thereby enabling the user to distribute the computations of complex virtual reality systems to multiple PCs. We chose the UDP over TCP protocol because it did not have the computational overhead associated with data checking such as handshaking, message acknowledgement, retransmission, or timeout. These checks on the data increase the reliability of the data transmissions but they also introduce transmission delays, making them inappropriate for real-time animations. The potential for loss of data in MSMS applications can be minimized by running them on a dedicated local network that connects PCs in short distances and carries only animation data for MSMS (see below).

To streamline the sending of animation data to MSMS, we have developed a protocol called "Feature Commands" that can be used by any program to send animation data to MSMS via UDP. A packet of animation data representing an animation frame must have the following format:

$$\{[\mathrm{ID}, \mathrm{F}, \mathrm{V}], [\mathrm{ID}, \mathrm{F}, \mathrm{V}], [\mathrm{ID}, \mathrm{F}, \mathrm{V}] \ldots\}$$

where ID is the unique identifier of an MSMS object such as a joint or segment, F is the specific feature of the object that must be modified such as its position, color, or size, and V is the new value of the feature. This simple protocol allows the users to animate the motion of a joint, modify the physical attributes of objects such as their color, size, and visibility, draw arbitrary trajectories in the VR scene, display text, or play sound at appropriate times during the animation. With the "Feature Command" protocol, the user must only send the data for those features that are to be animated, which minimizes the size of the packets to be sent via UDP. The forming of the packets for large models could be difficult and time consuming, although it has to be done only once. When a MSMS model is exported to Simulink, the packet for animation of all of its joints is created automatically. If other programs such as C are used to generate the "Feature Commands" for animation, they have to create their own packets following the guidelines in MSMS's user guide.

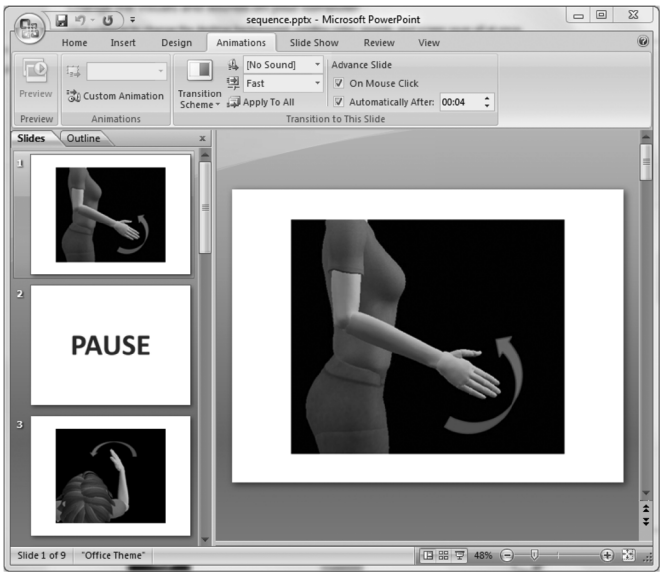


Fig. 4. Building an animation sequence with precise timing and order in PowerPoint. The order of the slides in PowerPoint presentation specifies the order in which primitive movements are animated. The PAUSE slide represents wait time between two primitive movements. For each slide representing a primitive movement or pause, "Automatically After" parameter is used to specify its duration.

III. Building Animation Sequences in PowerPoint

MSMS allows nonexpert users such as clinicians to build complex animations of the activities of daily living (ADL) in the familiar and easy to use environment of PowerPoint software. The process involves creating a library of motion files representing primitive movements (preferably by experts) and then arranging the order and properties of the images corresponding to the primitive movements in PowerPoint (by expert or nonexpert users) in any order to build customized animations of complex movements (Fig. 1).

The library of the primitive movements contain pairs of motion and image files representing specific movements such as elbow flexion, hand closing, etc. For example, the elbow flexion movement from full extension to 90° flexion must be represented by a MSMS motion file containing the motion data and an image file of the same name depicting the same movement. The motion files for primitive movements can be created using any method such as capturing the movement of a human subject performing the movement, motion synthesis, or physics-based simulations, as described above. The images depicting the primitive movements can be obtained by photography, drawing, or screen capture of the virtual model in MSMS window. The library also needs an additional image file named pause that can be used to represent pauses between the primitive movements.

Once the library of primitive movements is created, the motions in the library can be sequenced in any order in PowerPoint. The user has to create a new PowerPoint presentation and insert images (one image per slide) corresponding to the primitive movements or pauses (Fig. 4). The user can change the order of the slides to change the order of the animations. To repeat one motion more than once, the user can copy and paste the primitive movement's corresponding slide as many times as desired. Further, the user can edit the animation properties of slides to change the total animation time of each primitive motion and the length of pause between two primitive movements. To change the duration, the user simply modifies the value of "Automatically After" parameter in PowerPoint's animation menu (Fig. 4). MSMS uses this value to scale the timing of the primitive motion data so that the same motion animates slower or faster depending on the duration specified by the user and the duration of the original primitive motion. This allows the user to easily create new animation of ADL with the desired sequence, speed, and timing. The PowerPoint presentation file must then be saved in XML format (one of the save options in PowerPoint).

The PowerPoint presentation file in XML format is then parsed by MSMS to obtain the sequence and timing information and create a combined motion file that represents the whole sequence. The image names in the slides are used to find the corresponding primitive motion file, scale the timing of the motion data if necessary, and copy them to the combined motion file. The scaling simply calculates a new time step between the consecutive frames of motion data by dividing the animation time specified in the slide by the number of data frames in the primitive motion file. To represent a pause in the animation, the last frame of data from the preceding primitive motion file is repeated for the duration of the pause specified in the slide. The combined motion file has a format similar to the primitive motion files in the library and can be directly loaded and animated in MSMS. When parsing the PowerPoint file, MSMS also creates a sequence file. The sequence file has the same number of rows as the motion file. A row in the sequence file corresponds to a row in the motion file, specifying its type (animation or pause), and in the case of animation, the name of the primitive movement to which it belongs. The information in the sequence file is not required for loading and animation of the motion file but it is essential for interactive rehabilitation applications in which other patient data such as EMG of the muscles or movement of the patient's limbs must be recorded and synchronized with the animation data. For a specific application of this feature see Zeher *et al.* [29] and example applications of MSMS animations below.

IV. Real-Time and Nonreal-Time Applications

In nonreal-time applications, usually there are no interactions between MSMS animations and the outside world (subject and measurement devices) or there are interactions but slow response times and delays can be tolerated. For example, an engineer simulating the control of prosthetic limbs in Simulink does not have to animate the resulting movement in real-time. The resulting motion data can be stored in a motion file and animated later at a desired speed to evaluate the performance of the prosthetic control system. Alternatively, the resulting movement can be sent to MSMS via UDP for online animations. This allows the engineer to see the resulting movement while the movement is simulated. The animation will then run at the same speed as the physics-based simulation, which may be slower or faster than real-time depending on the complexity of the simulations and the available processing power. Nonreal-time animations can also be used in inspection of the motion data captured from human subjects, animation of correct motion patterns for demonstration or training, replay of motion data

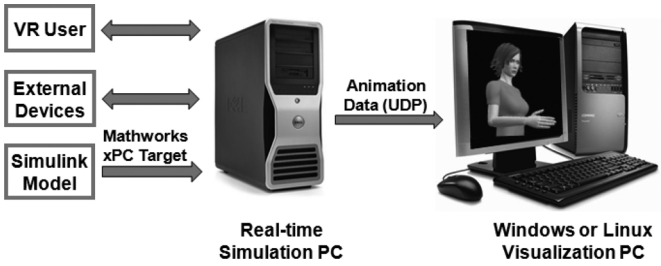


Fig. 5. Minimal setup for real-time applications composed of two PCs. The real-time simulation PC uses Mathwork's xPC Target real-time kernel to manage the interactions with the subject and external measurement devices, run the simulation code and application logic, and broadcast the motion data via UDP. The visualization PC runs MSMS under Windows or Linux operating systems to intercept the motion data sent from the real-time PC and use it to animate the virtual model as fast as possible. The motion data broadcast can be intercepted and visualized in multiple visualization PCs for different users (e.g., patient, operator, and clinician) from different perspectives.

recorded in scientific experiments, and any application where the animated motion does not have to be altered in response to or synchronized with the other real-time events.

Because of the lax requirement on response time, nonreal-time applications require minimal hardware setup, as little as a single PC with Windows or Linux operating systems. The single PC can interact with the outside world via its input/output ports, execute the simulations or programs that generate animation data, and run MSMS to animate the virtual model. The hardware configuration can be tailored to the application. For example, physics-based simulations would benefit from more powerful processors and larger memory while interactive virtual environments may use 3-D stereoscopic displays and head tracking sensors.

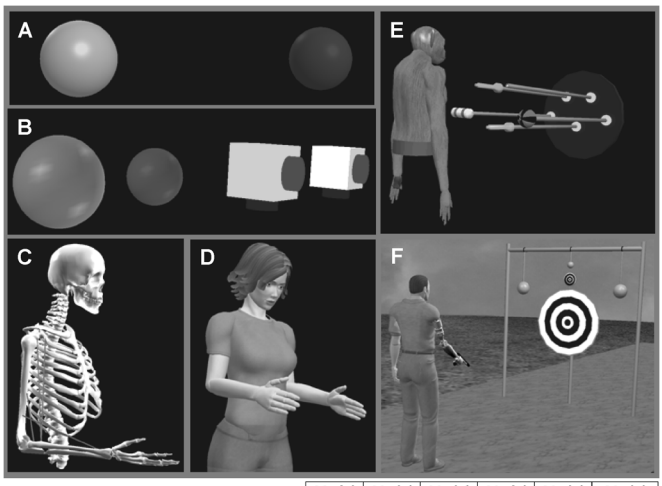
For real-time applications MSMS animation must interact with and respond to the inputs by the user and external devices. For the user to be immersed in and interact with the virtual world, the MSMS animations must be rendered at appropriate speed and with minimal delay. The minimal setup for real-time applications includes at least two PCs: a real-time PC and a visualization PC (Fig. 5). The real-time PC must run a real-time operating system to dedicate all its resources to running the simulations and the application logic, input/output from and to the human user and the external devices, and generation and transmission of animation data to the visualization PC via UDP [30]. The visualization PC, on the other hand, must run a Windows or Linux operating system and must dedicate its resources to animation of the virtual model.

Although other real-time processors can be used to control virtual applications, the real-time setup in Fig. 5 offers a number of advantages especially when it is used with MSMS animations. This setup uses xPC Target real-time kernel (Mathworks Inc.) that turns any PC into a powerful real-time system. Further, it allows the users to compile and download MSMS-generated Simulink models automatically to the real-time PC. As mentioned above, MSMS can convert its models into Simulink models that not only compute movement dynamics but also properly package and send the resulting motion data to MSMS for animation. The ability to automatically compile the Simulink model and immediately run it in real-time speeds up the development time of virtual applications. Separate PCs for simulation and visualization was first proposed by Hauschild *et al.* [30] and has since been duplicated in other laboratories (see below). The proposed setup includes also an operator PC that is not involved in the runtime operation of the virtual environment but can be used by the operator to develop virtual models and simulations, compile and download the simulation models to real-time PC, and view and control the operation of the virtual environment. For example, in the operator PC, the operator can view the same virtual environment presented to the subject from a different perspective.

Whether an animation can be run in real-time or not depends on many factors such as the complexity of the virtual model and the available processing power. For a given computing power and computational load, the user must be able to determine whether the virtual application can be executed in real-time or not. This knowledge enables the user to take appropriate action such as increasing the computational power or reducing computational load by simplifying the virtual model until real-time performance can be achieved. The xPC Target kernel in the real-time PC displays detailed execution time information to the user in run time indicating whether the application can be run in real-time or not. The visualization PC runs Windows or Linux, which are both nonreal-time operating systems. The user must configure these to give the highest priority to MSMS animations and close as many system programs as possible. Our experience shows that this is easier to do in Linux and, therefore, the MSMS models can be animated faster in Linux than in Windows. But MSMS users have been able to achieve real-time performance in both operating systems. To enable the user to determine whether MSMS models can be animated in real-time, MSMS offers a utility that can display the rendering statistics to the user while the model is animated. Comparison of these data with the required update rate can show whether the model can be animated in real-time or not.

To evaluate the effect of model complexity and processing power on MSMS rendering times, we animated a number of typical MSMS models with varying levels of complexity in four PC configurations (Fig. 6). The model complexity ranged from a simple two-sphere model used in nonhuman primate experiments to complex models used in rehabilitation and patient training. The simulated animation data were sinusoidal variations of all of the DOF in the model and were generated by a Simulink program running on the same PC. One thousand frames of motion data were sent via UDP at a high rate to ensure that there were always data in the buffer waiting to be rendered by MSMS. MSMS received and rendered as many frames of data as possible. The total time of animation was divided to the total number of frames actually rendered by MSMS to obtain the average rendering time per frame of data. Rendering times therefore represent the time spent in the visualization PC by MSMS to receive and unpack a UDP packet containing a frame of motion data, apply the data to the model, and render the new posture on the display. To estimate the overall latencies, one must also add the latencies associated with the processes in the real-time PC and the network transport that were not measured here.

The measured rendering times show that all setups tested here performed well on 2-D rendering of simple models but larger models can only be rendered using higher end video cards, especially if they have to be rendered in stereoscopic 3-D. Interestingly, MSMS rendering times did not always increase with



	Model A	Model B	Model C	Model D	Model E	Model F
PC1 - Windows 7, 8 GB RAM, 3.2 GHz i5-650 CPU with Integrated Intel HD graphics	2.13	1.54	11.22	40.15	31.35	100.50
PC2 - Windows XP, 3 GB RAM, 3.6 GHz Xeon CPU, NVIDIA Quadro FX3450 graphics	14.9	13.35	13.27	15.20	15.75	38.23
PC3 - Windows 7, 24 GB RAM, 2x3.33 GHz Xeon CPU, ATI FirePro V4800 graphics	14.35	10.02	3.46	7.86	7.87	10.62
PC4 - Windows 7, 24 GB RAM, 2x3.33 GHz Xeon CPU, 3D Stereo NVIDIA Quadro FX4800	2.65	9.59	8.34	12.7	9.40	16.72
PC4 - Stereoscopic Rendering	14.35	14.61	9.12	22.73	16.80	17.91

Fig. 6. Renderings times in milliseconds for typical MSMS models in four PC configurations. Models vary in complexity from Model A containing two objects and 6 degrees-of-freedom (commonly used in cortical control experiments with nonhuman primates) to Model F, a physics-based shooting game containing 48 objects and 69 degrees-of-freedom. Rendering times for stereoscopic visualizations are provided for PC4, the only PC with stereo-capable graphics.

model complexity. This might be a feature of the Java3D technology employed by MSMS. Similar observations were made by Burdea and Coiffet [31], who noticed little change in Java3D rendering times over a wide range of model sizes. We should also note that, for the measurements here, we did not make any special modifications to the PCs. Running the Simulink program with MSMS on the same PC could slow down rendering times. To minimize the latencies, the visualization PC must be carefully configured so that it gives the highest priority to MSMS and does not run any other applications or services. The Windows rendering times reported here can be improved significantly by the use of Linux operating system that allows more user control over its resources [32].

V. Applications of MSMS Animations

MSMS provides flexible and powerful tools for building virtual models of arbitrary complexity and animating them with motion data from offline and online sources. MSMS models and animations can be turned into interactive virtual applications using PCs and Matlab toolboxes that are commonly available in most academic institutions and research laboratories. Therefore, researchers and developers with common computer skills can develop complete virtual applications in readily available PC hardware.

In our laboratory, we have used MSMS animation tools in interactive virtual reality environments to develop and evaluate neural prosthesis control systems for paralyzed patients and amputees [33]. In these applications, the subject's voluntary command signals such as the movement of the intact joints are used to control the movement of a simulated limb in virtual world. The real-time stereoscopic display of the resulting movement from the subject's perspective provided the visual feedback that enabled the user to see the consequence of his/her voluntary actions and adjust them if necessary. These environments therefore enable us to evaluate the feasibility of a neural prosthetic control system before clinical deployment and train the patients to operate their prostheses effectively.

MSMS animation tools have also been used in many other laboratories. For example, as part of a large multi-center project headed by The Johns Hopkins Applied Physics Laboratory, a virtual integration environment was developed to streamline the collaborative development of sophisticated neural prostheses for upper limb amputees [34]. The virtual environment uses a real-time hardware setup similar to that in Fig. 5 to acquire and process neural commands such as those from motor cortex or EMG of residual muscles, driving the movement of a simulated prosthetic arm modeled and animated in MSMS. This virtual environment enabled system integrators to test and evaluate quantitatively the performance of neural control strategies developed in different laboratories on the same engineering test bed.

In Walter Reed Army Medical Center, Zeher *et al.* [29] have used MSMS's animation tools to develop a virtual version of "Mirror Therapy" that has been shown to alleviate amputee phantom limb pain [35]. They have synthesized a library of primitive arm and hand movements that can be configured in any order and timing in PowerPoint by nonexpert users such as clinicians. The resulting animation sequences are then presented to the amputees who are asked to attempt the same movements with their phantom limb. The animations presented to the patient and the resulting changes in the EMG of residual muscles in the stump must be recorded and synchronized so that their relationships can be studied to assess treatment outcomes. To synchronize these data, they used a setup similar to that in Fig. 5 where the real-time PC reads in the MSMS-generated motion and sequence files for ADL and sends the desired animation frames to the visualization PC while simultaneously measuring the patient's EMG data.

Motor control and learning is another important area of application where human or nonhuman primate subjects interact with MSMS generated animations. Markus Hauschild used the architecture for human-in-the-loop simulations that he developed at the University of Southern California [30] to develop a virtual simulation environment to conduct closed-loop cortical control experiments with nonhuman primates at Caltech. In experiments that are now routine in their laboratory, they use the Windows operating system to construct and animate MSMS models of 3-D reaching movements to virtual targets [36], [37]. In their configuration with Windows XP, the measured latencies for streaming the motion data from the real-time PC to receiving the motion data and rendering the virtual model was 15–20 ms (personal communication), which was adequate for their closed-loop experiments with nonhuman primates. At Stanford University, Cunningham *et al.* [32] developed a closed-loop simulator for investigation of the role of feedback control in brain–machine interfaces. In their simulator, a human or nonhuman primate subject can interact with simulated objects that are modeled in and animated by MSMS and presented in 3-D stereo-

scopic displays. The development of the closed-loop simulator was motivated by the finding that the commonly used methods for offline development and testing of neural decode algorithms neglect the potential for a real neural prosthesis user to make corrections to control strategies to improve performance. Their virtual environment had a stringent real-time performance requirement that was difficult to achieve under Windows operating system. Switching to Linux, which has less overhead and more control over the system's resources, enabled them to achieve faster rendering times for MSMS animations. The measured latency and jitter in their whole system was 7 ± 4 ms, which was critically important for their study [32]. In a collaborative project between Johns Hopkins and Rochester Universities, Aggarwal *et al.* [27], integrated MSMS models and animations in a virtual environment to perform closed-loop experiments with nonhuman primates. In their application, MSMS is being used to build complex virtual models of arm, hand, and objects to study neural decoding algorithms for dexterous control of reach and grasp movements. In the authors' opinion, they can use virtual multi-joint arm models in MSMS in place of physical prototypes that are costly to develop and maintain. Further, in the virtual environment, the predicted arm movements from the neural decode algorithms can be easily overlaid with the actual limb movement obtained from motion capture system, providing a rapid validation of the newly developed cortical decode algorithms.

VI. Conclusion

MSMS provides its users with interactive tools to model arbitrarily complex models of biological and prosthetic limbs and the objects in the task environment with which they interact. These models can then be turned into simulation models in Simulink to perform physics-based simulation of the limb's movement or produce motion data to animate the MSMS model. The combination of a real-time PC for execution of the application logic and external interfaces and a visualization PC for animation and stereoscopic display of the virtual models is a powerful tool for development of new virtual applications for research and treatment of movement disorders. The already deployed applications in major national laboratories are enabling investigators to understand the control of human movement and develop innovative treatments for movement disorders.

References

[1] R. Davoodi and B. J. Andrews, "Optimal control of FES-assisted standing up in paraplegia using genetic algorithms," *Med. Eng. Phys.*, vol. 21, pp. 609–617, 1999.

[2] R. F. Kirsch, A. M. Acosta, and D. Yu, "Feasibility of restoring shoulder and elbow function in high tetraplegia by functional neuromuscular stimulation," in *Proc. 20th IEEE Eng. Med. Biol. Soc. Conf.*, 1998, pp. 2602–2604.

[3] R. F. Kirsch, A. M. Acosta, F. C. T. van der Helm, R. J. J. Rotteveel, and L. A. Cash, "Model-based development of neuroprostheses for restoring proximal arm function," *J. Rehabil. Res. Developm.*, vol. 38, no. 6, pp. 619–626, Nov. 2001.

[4] T. A. Kuiken, G. L. Li, B. A. Lock, R. D. Lipschutz, L. A. Miller, K. A. Stubblefield, and K. B. Englehart, "Targeted muscle reinnervation for real-time Myoelectric control of multifunction artificial arms," *JAMA*, vol. 301, no. 6, pp. 619–628, Feb. 2009.

[5] A. B. Schwartz, "Cortical neural prosthetics," *Annu. Rev. Neurosci.*, vol. 27, pp. 487–507, 2004.

[6] F. Multon, R. Kulpa, L. Hoyet, and T. Komura, "Interactive animation of virtual humans based on motion capture data," *Comput. Animat. Virtual Worlds*, vol. 20, pp. 491–500, 2009.

[7] M. Kasap, P. Choudhuri, and N. M. Thalmann, "Fast EMG-data driven skin deformation," *J. Vis. Comput. Animat.*, vol. 20, pp. 153–161, 2009.

[8] F. Multon, R. Kupla, and B. Bideau, "MKM: A global framework for animating humans in virtual reality applications," *Presence*, vol. 17, no. 1, pp. 17–28, 2008.

[9] L. Eriksson, F. Sebelius, and C. Balkenius, "Neural control of a virtual prosthesis," in *Perspectives in Neural Computing: Proceedings of ICANN '98*, L. Niklasson, M. Bodén, and T. Ziemke, Eds. Berlin, Germany: Springer-Verlag, 1998.

[10] F. Sebelius, L. Eriksson, C. Balkenius, and T. Laurell, "Myoelectric control of a computer animated hand: A new concept based on the combined use of a tree-structured artificial neural network and a data glove," *J. Med. Eng. Technol.*, vol. 30, no. 1, pp. 2–10, Jan. 2006.

[11] A. Soares, A. Andrade, E. Lamounier, and R. Carrijo, "The development of a virtual Myoelectric prosthesis controlled by an EMG pattern recognition system based on neural networks," *J. Intell. Inf. Syst.*, vol. 21, no. 2, pp. 127–141, 2003.

[12] A. C. Dupont and E. L. Morin, "A Myoelectric control evaluation and trainer system," *IEEE Trans. Rehabil. Eng.*, vol. 2, no. 2, pp. 100–107, Jun. 1994.

[13] D. Nishikawa, W. Yu, H. Yokoi, and Y. Kakazu, "EMG prosthetic hand controller using real-time learning method," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1999, pp. I-153–I-158.

[14] M. Kuttuva, G. Burdea, J. Flint, and W. Craelius, "Manipulation practice for upper-limb amputees using virtual reality," *Presence*, vol. 14, no. 2, pp. 175–182, 2005.

[15] J. L. Pons, R. Ceres, E. Rocon, S. Levin, I. Markovitz, B. Saro, D. Reynaerts, W. V. Moorleghem, and L. Bueno, "Virtual reality training and EMG control of the MANUS hand prosthesis," *Robotica*, vol. 23, pp. 311–317, 2005.

[16] L. Piron, P. Tonin, F. Piccione, V. Laia, E. Trivello, and M. Dam, "Virtual environment training therapy for arm motor rehabilitation," *Presence*, vol. 14, no. 6, pp. 732–740, 2005.

[17] D. Jack, R. Boian, A. S. Merians, M. Tremaine, G. C. Burdea, S. V. Adamovich, M. Recce, and H. Poizner, "Virtual reality-enhanced stroke rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 9, no. 3, pp. 308–318, Sep. 2001.

[18] M. Kuttuva, R. Boian, A. Merians, G. Burdea, M. Bouzit, J. Lewis, and D. Fensterheim, "The Rutgers Arm, a rehabilitation system in virtual reality: A pilot study," *Cyberpsychol. Behav.*, vol. 9, no. 2, pp. 148–151, Apr. 2006.

[19] S. Zhang, H. Hu, and D. Gu, "A data-driven 3-D animation system for tele-rehabilitation," in *Proc. IEEE Int. Conf. Mechatron. Automat.*, Harbin, China, 2007, pp. 2166–2171.

[20] S. Zhang, H. Hu, and H. Zhou, "An interactive Internet-based system for tracking upper limb motion in home-based rehabilitation," *Med. Biol. Eng. Comput.*, vol. 46, pp. 241–249, 2008.

[21] C. B. Lourenco, L. Azeff, H. Sveistrup, and M. F. Levin, "Effect of environment on motivation and sense of presence in healthy subjects performing reaching tasks," *Proc. Virtual Rehabil.*, pp. 93–98, 2008.

[22] M. K. Holden, T. A. Dyar, L. Schwamm, and E. Bizzi, "Virtual-environment-based telerehabilitation in patients with stroke," *Presence*, vol. 14, no. 2, pp. 214–233, 2011.

[23] S. Baek, S. Lee, and G. J. Kim, "Motion retargeting and evaluation for VR-based training of free motions," *Vis. Comput.*, vol. 19, pp. 222–242, 2003.

[24] G. H. Mulliken, S. Musallam, and R. A. Andersen, "Decoding trajectories from posterior parietal cortex ensembles," *J. Neurosci.*, vol. 28, no. 48, pp. 12913–12926, Nov. 2008.

[25] G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz, "Control of a brain-computer interface without spike sorting," *J. Neural Eng*, vol. 6, no. 5, p. 055004, Oct. 2009.

[26] S. P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, and M. J. Black, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," *J. Neural Eng.*, vol. 5, no. 4, pp. 455–476, Dec. 2008.

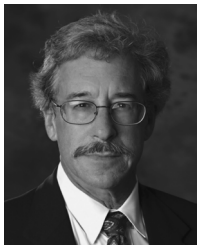
[27] V. Aggarwal, M. Kerr, A. G. Davidson, R. Davoodi, G. E. Loeb, M. H. Schieber, and N. V. Thakor, "Cortical control of reach and grasp kinematics in a virtual environment using musculoskeletal modeling software," in *Proc. IEEE EMBS Conf. Neural Eng.*, 2011, pp. 388–391.

[28] R. Davoodi, C. Urata, M. Hauschild, M. Khachani, and G. E. Loeb, "Model-based development of neural prostheses for movement," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 11, pp. 1909–1918, Nov. 2007.

[29] M. J. Zeher, R. S. Armiger, J. M. Burck, C. Moran, J. B. Kiely, S. R. Weeks, J. W. Tsao, P. F. Pasquina, R. Davoodi, and G. Loeb, "Using a virtual integration environment in treating phantom limb pain," *Stud. Health Technol. Inform.*, vol. 163, pp. 730–736, 2011.
[30] M. Hauschild, R. Davoodi, and G. E. Loeb, "A virtual reality environment for designing and fitting neural prosthetic limbs," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 15, no. 1, pp. 9–15, Mar. 2007.
[31] G. C. Burdea and P. Coiffet, *Virtual Reality Technology*, 2nd ed. Hoboken, NJ: Wiley, 2003.
[32] J. P. Cunningham, P. Nuyujukian, V. Gilja, C. A. Chestek, S. I. Ryu, and K. V. Shenoy, "A closed-loop human simulator for investigating the role of feedback-control in brain-machine interfaces," *J. Neurophysiol.*, vol. 105, no. 4, pp. 1932–1949, Apr. 2011.
[33] R. Kaliki, R. Davoodi, and G. E. Loeb, "Evaluation of non-invasive command scheme for upper limb prostheses in a virtual reality reach and grasp task," *IEEE Trans. Biomed. Eng.*, to be published.
[34] W. Bishop, R. Armiger, J. Burck, M. Bridges, M. Hauschild, K. Englehart, E. Scheme, R. J. Vogelstein, J. Beaty, and S. Harshbarger, "A real-time virtual integration environment for the design and development of neural prosthetic systems," in *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, 2008, vol. 2008, pp. 615–619.
[35] B. L. Chan, R. Witt, A. P. Charrow, A. Magee, R. Howard, P. F. Pasquina, K. M. Heilman, and J. W. Tsao, "Mirror therapy for phantom limb pain," *N. Eng. J. Med.*, vol. 357, no. 21, pp. 2206–2207, Nov. 2007.
[36] M. Hauschild, G. H. Mulliken, G. E. Loeb, and R. A. Andersen, "Decoding instantaneous hand position from posterior parietal cortex in a natural task," in *Proc. Annu. Meet. Soc. Neurosci.*, 2008.
[37] M. Hauschild, G. H. Mulliken, G. E. Loeb, and R. A. Andersen, "Adaptation of posterior parietal cortex to visually imposed perturbations of movement dynamics," in *Proc. Annu. Meet. Soc. Neurosci.*, 2009.

Rahman Davoodi received the B.S. degree in mechanical engineering and M.Sc. degree in biomechanical engineering both from Sharif University of Technology, Tehran, Iran, and the Ph.D. degree in biomedical engineering from University of Alberta, Edmonton, AB, Canada.

He is currently a Research Assistant Professor in Department of Biomedical Engineering, University of Southern California, Los Angeles. His current research is focused on model-based development and clinical fitting of motor neural prostheses for paralyzed and amputee patients. He has developed functional electrical stimulation (FES) systems to restore normal movements to paralyzed patients and enable them to improve their cardiovascular fitness by FES-assisted exercise. More recently, he has led the development of software tools for modeling and simulation of complex human and prosthetic limbs that are used in many laboratories around the world to study the motor control of movement by the central nervous system, virtually prototype various stages of development and clinical fitting of motor neural prostheses, and develop virtual rehabilitation therapies for patients with variety of movement disorders.



Gerald E. Loeb (M'98–SM'07) received the B.A. degree and the M.D. degree from Johns Hopkins University, Baltimore, MD, in 1969 and 1972, respectively, and did one year of surgical residency at the University of Arizona, Tucson, before joining the Laboratory of Neural Control at the National Institutes of Health (1973–1988).

He was Professor of Physiology and Biomedical Engineering at Queen's University in Kingston, ON, Canada (1988–1999) and is now Professor of Biomedical Engineering and Director of the Medical Device Development Facility at the University of Southern California (USC), Los Angeles. He was one of the original developers of the cochlear implant to restore hearing to the deaf and was Chief Scientist for Advanced Bionics Corporation (1994–1999), manufacturers of the Clarion cochlear implant. He is a holder of 52 issued U.S. patents and author of over 200 scientific papers. Most of his current research is directed toward sensorimotor control of paralyzed and prosthetic limbs. His research team developed BION injectable neuromuscular stimulators and has been conducting several pilot clinical trials. They are developing and commercializing a biomimetic tactile sensor for robotic and prosthetic hands through a startup company for which he is Chief Executive Officer. His lab at USC is developing computer models of musculoskeletal mechanics and the interneuronal circuitry of the spinal cord, which facilitates control and learning of voluntary motor behaviors by the brain. These projects build on his long-standing basic research into the properties and natural activities of muscles, motoneurons, proprioceptors, and spinal reflexes.

Dr. Loeb is a Fellow of the American Institute of Medical and Biological Engineers.