**Rahman Davoodi***
**Gerald E. Loeb**
Department of Biomedical Engineering
University of Southern California
Los Angeles, California 90089

# Development of a Physics-Based Target Shooting Game to Train Amputee Users of Multijoint Upper Limb Prostheses

## Abstract

For upper limb amputees, learning the control of myoelectric prostheses is difficult and challenging. Introduction of newer prostheses with multiple degrees of freedom controlled by various neural commands will make such training even more difficult. To produce smooth and human-like movements, the user must learn to produce multiple neural commands with precise amplitude and timing. To aid in training of the amputee users, we have developed a realistic and motivating virtual environment (VE) consisting of a physics-based target shooting game. The users' neural commands such as EMG, cortical neural activity, or voluntary movements of the residual limbs can be used to control the movement of a simulated prosthesis to point and shoot at virtual targets. In addition to the visual, sound, and performance feedback of the resulting movement, the game provides reaction forces in contact points that can be used to drive haptic displays. The timing measurements show that the physics-based simulation and rendering can be executed in real time in readily available PC systems. The target shooting game was developed in musculoskeletal modeling software (MSMS) that has been developed in our laboratory and is freely available for development of similar virtual training applications.

## 1 Introduction

Compared to the human arm, the mechanical design and control of currently available prosthetic arms are primitive. The limited degrees of freedom reduce the possible movements by the prosthetic arm and the primitive control strategies produce unnatural, robot-like movements. Currently, the users of myoelectric prostheses are limited to a control strategy that allows them to control one joint at a time. To grasp an object, for example, the patient may successively activate different sets of residual muscles to rotate the forearm, to flex the wrist, and then close the hand, which produces slow and unnatural movements. The control strategy may appear to be simple, but in practice, it is difficult to master and tedious to execute. The patient must learn to produce noisy EMG signals from muscles that are not naturally related to the controlled joints. Furthermore, patients have no feedback from the myoelectric signals they generate, to know whether these signals are appropriate until the limb moves visibly

Presence, *Vol. 21, No. 1, Winter 2012,* 85–95


*Correspondence to davoodi@usc.edu.

(Scoufield & Schreiner, 2004). Myoelectric training therefore tends to be difficult, time-consuming, and stressful, even for relatively primitive prosthetic systems.

The training difficulty can be expected to increase exponentially as the number of mechanical degrees of freedom and the corresponding sources of neural commands increase. For example, the targeted reinnervation procedure provides multiple sources of EMG commands for control of multijoint prostheses (Kuiken et al., 2009), but the patients must receive extensive training before and after receiving the actual arm. Even more ambitious research aims to use the command signals from multichannel electrodes implanted in the brain's motor areas to directly control the movement of multijoint prostheses (Mulliken, Musallam, & Andersen, 2008; Fraser, Chase, Whitford, & Schwartz, 2009; S. P. Kim, Simeral, Hochberg, Donoghue, & Black, 2008).

Virtual training environments offer many advantages for training the amputee patients and have been used for training in other complex systems such as the use of flight simulators to teach the operation of an airplane, training human or primate subjects to perform motor tasks (Nikooyan & Zadpoor, 2009; Kuhlen, Kraiss, & Steffan, 2000; Mulliken et al., 2008; S. P. Kim et al., 2008), training and rehabilitation of patients with neurological disorders (Piron et al., 2005; Jack et al., 2001; Holden, Dyar, Schwamm, & Bizzi, 2011), training the control of paralyzed limbs by functional electrical stimulation (Durfee, Mariano, & Zahradnik, 1991; Davoodi & Andrews, 1998; Kirsch, Acosta, van der Helm, Rotteveel, & Cash, 2001; Riener & Fuhr, 1998), and training amputee patients to operate their myoelectric prostheses (see Section 2, Related Work). Computer simulations can represent scenarios that are difficult or impossible to create in real experiments. Perturbations such as changing the size or shape of objects can be introduced instantly; and the task difficulty and the type and modality of the user feedback can be enhanced gradually to improve learning without being limited by the constraints and expense of physical apparatus. Importantly, virtual training can be used in the early stages of recovery from amputation when training is critically important (Pons et al., 2005; Malone et al., 1984) but cannot be delivered using physical limbs.

Here we report on the development of a physics-based target shooting game as a realistic, motivating, and engaging environment to train the upper limb amputees to operate their multijoint neural prostheses. The game was developed using freely available software tools developed in our laboratory that have been described elsewhere (Davoodi, Urata, Hauschild, Khachani, & Loeb, 2007; Davoodi & Loeb, 2011). The particular training game was designed to identify general requirements and demonstrate capabilities that can be retasked readily using this application development tool.

## 2 Related Work

A number of studies have used simulated virtual prostheses to evaluate the viability of novel prosthetic concepts and train amputee patients to operate them effectively. For example, Eriksson, Sebelius, and Balkenius (1998) used an animated hand to investigate the feasibility of EMG control of a prosthetic hand with multiple degrees of freedom. In a study by Sebelius, Eriksson, Balkenius, and Laurell (2006), 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 (VE). Such virtual training environments enable the investigators to test the viability of novel prosthetic concepts and refine them before they are actually manufactured. A similar approach was used by Soares, Andrade, Lamounier, and Carrijo (2003), in which a human arm with a realistic appearance was used to make a visual presentation of the postures of a prosthetic arm corresponding to the patient's EMG commands. Dupont and Morin (1994) 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 15 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. Nishikawa, Yu, Yokoi, and Kakazu (1999) demonstrated the efficacy of real-time training in VEs

where they used primitive animations of the prosthetic hand in place of the real prosthesis. The trained motions were the desired postures the patients must achieve, however, and did not include dynamic movements or interactions with objects in the task environment. Smith, Huberdeau, Tenore, and Thakor (2009) developed a myoelectric decoding algorithm for continuous on-line decoding of finger movements. To help the subject produce the appropriate EMG commands, virtual multijoint prostheses in MSMS (Davoodi et al., 2007) were used to display the decoded finger movements to the subject in real time. Bouwsema, Sluis, and Bongers (2010) compared myoelectric signal training for opening and closing of the hand with a virtual trainer, an isolated prosthetic hand, and a prosthetic simulator. The trainees had to learn to produce the EMG commands for simple on/off control of the hand opening/closing. For this simple training task, they found that virtual training was as effective as the physical simulators and there was no difference in training methods.

The above studies have all used animation-only VEs where realistic physics-based movements and interactions have been ignored. But the training tasks in these studies were simple and included the production of discrete commands to successively control few degrees of freedom. Learning to continuously and simultaneously control multiple degrees of freedom in a more sophisticated prosthesis, however, is a more difficult task and it remains to be seen whether this training can be performed effectively by animation-only training environments. It has been argued that the transfer of skills from virtual training to real-world training can be maximized if the virtual task simulates both the appearance and the dynamics of the real world with higher fidelity and realism (Rose, Attree, Brooks, Parslow, & Penn, 2000; Lourenco, Azeff, Sveistrup, & Levin, 2008; Nikooyan & Zadpoor, 2009). The realism of simulations is especially important for training amputees, because nonrealistic movements are easy to detect and can negatively affect the sense of presence and immersion. Moreover, if the prosthesis does not behave realistically and violates the physical constraints of the real limb, the trainee will learn an incorrect control strategy that is not appropriate for the control of the real limb.

## 3 Methods

The architecture and computing environment of the target shooting game are shown in Figure 1. The purpose of the game is to help the amputee user to practice and learn the control of a multijoint prosthesis in a closed-loop virtual simulation environment.

To play the game, the amputee user must generate voluntary command signals from one or more of the available command sources to control the movement of the virtual arm and the index finger to point and shoot at the virtual targets. To facilitate learning, the physics-based movement of the virtual arm and target objects, sound, haptic feedback, and the resulting score are displayed to the trainee. The simulation and visualization are executed on separate PCs to ensure that the complex simulation and rendering can be performed in real time (Hauschild, Davoodi, & Loeb, 2007). The simulation PC is a regular PC that runs xPC Target real-time Kernel (Mathworks, Inc.) and is responsible for acquiring and decoding the user's voluntary command signals, simulation of movement and object interactions, execution of game control logic, driving haptic actuators, and sending the visual and sound feedback data to the visualization PC via UDP. The simulation code was first developed and tested in Simulink (Mathworks, Inc.), when runs under Windows and then it was compiled and downloaded into the Simulation PC for real-time execution. The visualization PC runs on the Windows (or Linux) operating system and is responsible for rendering and 3D stereoscopic visualization of movement from the user's perspective and display of the sound and performance data to the user.

## 4 Virtual Models of Amputee User, Prosthetic Limb, and Objects

The virtual model consisting of the amputee subject, prosthetic limb, handgun, targets, and the environment (see Figure 2) was developed using MSMS software (Davoodi et al., 2007; Davoodi & Loeb, 2011).

The model of the amputee subject represents an average male with shoulder disarticulation amputation (height 180 cm; weight 75 kg). The sizes and inertial

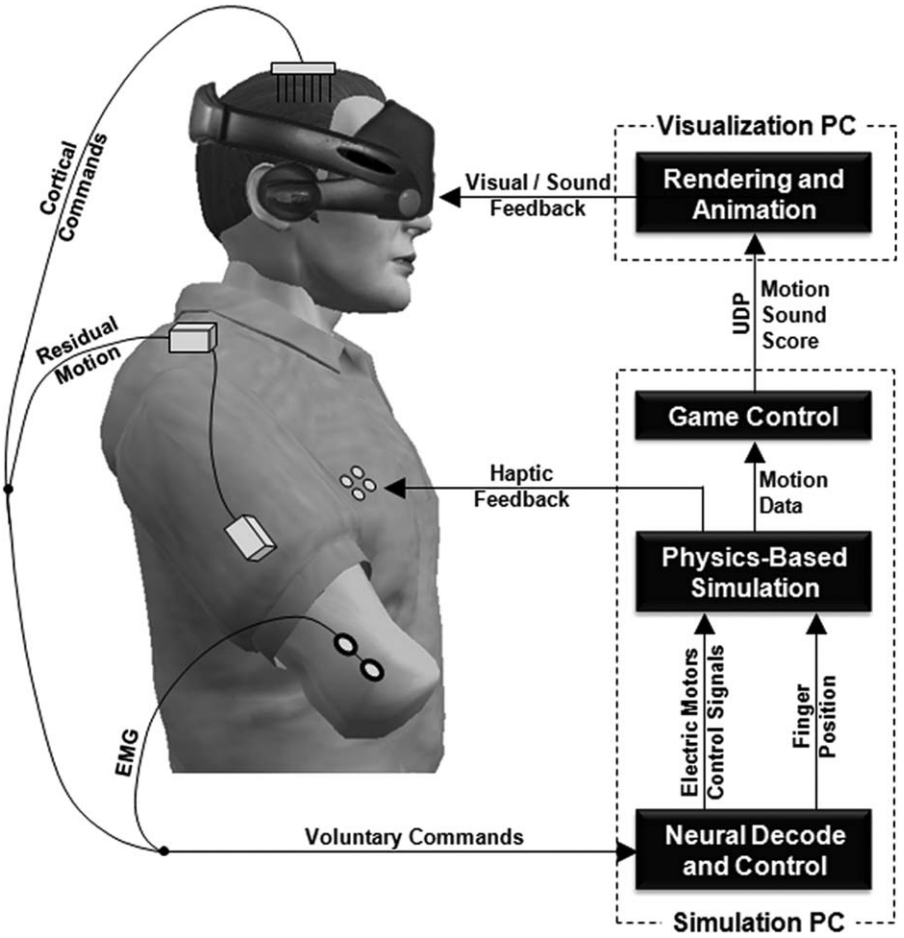


**Figure 1.** *Architecture, computing environment, and the user interface of the target shooting game.*

properties of the segments for the average male subject were obtained from anatomical tables (Winter, 1990). In MSMS, any number of primitive shapes or 3D meshes can be used to represent the appearance of each segment. For the amputee model, the images of the segments were exported from the human animation software Poser (Smith Micro, Inc.) in the form of 3D meshes with textures. These 3D meshes were then imported into MSMS and overlaid on the segments of the amputee model.

The model of the prosthetic limb was designed in SolidWorks (SolidWorks Corp.) and represents a real limb with 26 DOF (Bridges et al., 2010). The SolidWorks model of the prosthetic limb was automatically imported into MSMS and attached to the stump of the amputee model. The automatically imported model is an exact replica of the SolidWorks model in terms of appearance, linkage, and mechanical properties. The actuation motors for each joint were modeled in MSMS.

The gun was modeled as a segment that is attached to and moves with the prosthetic hand. Multiple primitive shapes were attached to this segment to form the appearance of a simple handgun. The trigger was attached to the gun via a pin joint allowing it to rotate about the axis of the pin joint. The bullet was modeled as a spherical segment that is attached to the ground via a 6-DOF joint. This joint allows the bullet to move and rotate in any direction with respect to the ground.

The target set is a combination of stationary and moving targets. The base of the target set is attached to the ground via a 6-DOF joint allowing the whole target set to be placed at any position or orientation within the

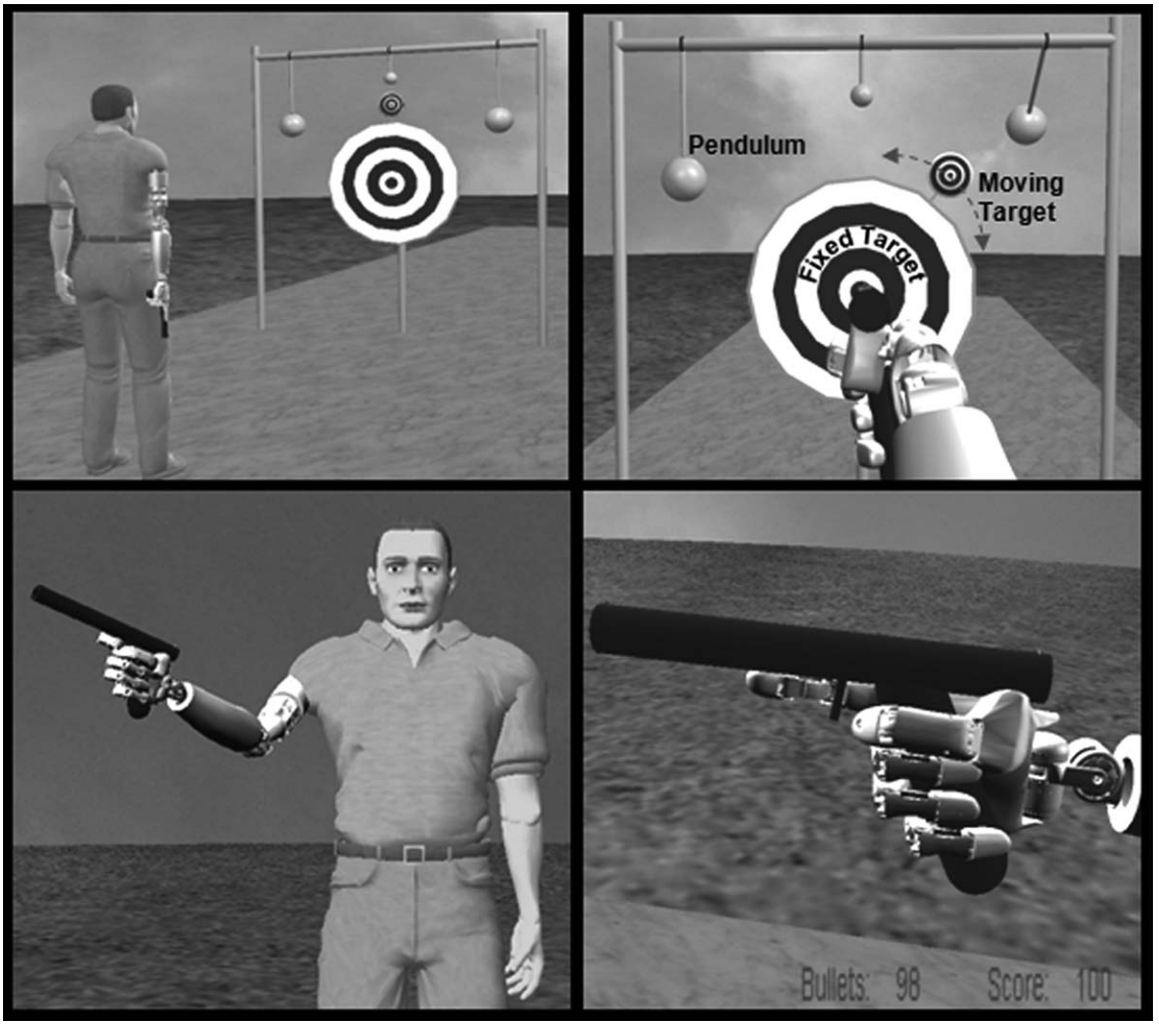


**Figure 2.** *Virtual model of the game in MSMS. The trainee can aim and shoot at the fixed target, moving target, or one of the three pendulums.*

environment. A circular plate with rings represents a target that is fixed to the base. A smaller moving target with rings is attached to the base via a pin joint, allowing it to move in a circular path around the fixed circular target. Three pendulums with varying sizes are attached to the base frame via pin joints, allowing them to swing toward and away from the user when hit by the bullets.

## 5 Physics-Based Simulation of Movement and Collisions

The virtual model of the game was exported into Simulink using MSMS's automatic export utility. It included the Newtonian equations of motion for the amputee user, the prosthetic limb, and the objects that were all modeled using SimMechanics toolbox (Mathworks, Inc.). But the exported model did not include the algorithms for decoding user commands, simulation of object collisions, and game control logic that were added manually as described below. The Simulink model can be numerically integrated in Simulink (during development and testing) and in real-time xPC Target PC (in the deployment stage for patient training) to predict the realistic physics-based motion of the virtual model in response to user commands and external forces such as gravity, friction, and collision forces.

#### 5.0.1 Simulation of Amputee User's Movement.

Because the prosthetic limb is attached to the end of the stump in the residual limb, its movement is affected by the movement of the stump. Therefore, any user movement that contributes to the movement of the stump must be simulated. To simulate amputee user's movements, the joints of the amputee model were forced (using SimMechanics's motion actuators) to follow the movement of their corresponding joints in the amputee user. The motion of the amputee user's joints must therefore be captured and used to drive the movement of the joints in a virtual amputee model. A complete motion capture of the amputee subject can allow the subject to freely move in the environment while playing the game. But if the amputee subject is confined to sitting on a chair and the trunk movement is restrained, the measurements could be limited to that of the stump and the head. The latter is used to control the camera's position and orientation to display the scene from the user's perspective.

#### 5.0.2 Simulation of Prosthetic Limb Movement.

The joints of the prosthetic limb were actuated by electric motors that were modeled in a C program embedded in a Simulink S-Function. The full dynamics of the electric motors were computationally demanding and had to be simplified to enable real-time execution. In the simplified model, the motor inductance was ignored and the transmission was assumed to be rigid. These had little effect on the simulation outputs but resulted in a model that was easier to integrate numerically and therefore could be simulated in real time.

#### 5.0.3 Simulation of Bullet and Targets Movement.

The bullet was simulated as a projectile that leaves the gun with a known initial velocity in the direction of the gun barrel at the time of shooting. From that point on, the movement of the bullet is affected only by gravity and the collision forces with the targets (see Section 5.0.4). The fixed circular target was kept in its initial position. The moving target was moved about its pin joint axis at a constant angular velocity. The direction of movement and the magnitude of speed were determined by the history of collisions with the bullet (see Section 5.0.4). The movements of the pendulums were simulated by solving the respective equations of motion for a single pendulum subject to the forces of gravity and collision with the bullet.

#### 5.0.4 Simulation of Collision and Contact.

Two types of collisions and contacts were modeled: the contact between the index finger and the trigger, and the collision between the bullet and the targets.

A spring model was used to simulate the contact between the finger and the trigger. SimMechanics' position sensors were attached to the fingertip and the trigger to obtain their current positions. When the fingertip position reached the resting position of the trigger, contact between them was established. From that point on, the trigger followed the movement of the fingertip; if the trigger moved beyond a threshold, the gun fired a single bullet. The finger must be extended back beyond the resting point of the trigger to disengage before another bullet can be fired. When the fingertip was in contact with the trigger, the displacement of the trigger from its resting position and its spring constant determined the contact force between the trigger and the fingertip. The simulated finger contact force was applied to and resisted the finger's simulated motion and can be used to drive haptic feedback devices such as electrotactile or vibrotactile stimulators placed on the skin of the residual limb (K. Kim, Colgate, Santos-Munne, Makhlin, & Peshkin, 2010; Nohama, Lopes, & Cliquet, 1995). Such haptic feedback enables the patient to feel and therefore regulate the level of finger contact force more effectively.

The collisions between the bullet and the targets were detected and handled in two stages. First, the existence of the collision between the bullet and the targets was checked. When there was no collision, the bullet and the moving targets continued their current movements. When the bullet collided with the fixed circular target, the point of collision was marked to reveal what part of the target was hit. When the moving circular target was hit, the target changed the direction of its angular

motion and increased its velocity by a percentage, making it harder for the trainee to target it the next time. When the bullet hit a pendulum target, an elastic collision formulation based on the conservation of momentum and kinetic energy was used to calculate the velocities of the bullet and the pendulum after the collision (Van Verth & Bishop, 2004).

A number of modifications and customizations were introduced to reduce the simulation times and to enable the successful detection and handling of collisions between the fast-moving bullet and the targets. Because the targets were all confined to a limited space, the collision between the bullet and the targets' bounding box (a box large enough to contain all targets) was checked first. The collisions with individual targets were checked only if a collision with the bounding box was detected. When a collision with a target was detected, no further collision checks were made in that time step. Because of the placement of the targets, it was impossible for the bullet to collide with more than one target at a time. To detect collision with targets, we initially checked the existence of penetration between the geometric shapes of the bullet and the targets. This proved inadequate because a fast-moving bullet could travel a long distance in a single time step from one side of the target to the other side, resulting in collisions that go undetected. A possible solution was to significantly reduce the simulation time step but the required time steps were then too small for real-time execution. Therefore, a different solution was adopted that checked for the collision between a line segment (from the bullet's current position to its predicted position at the end of a single time step) and the geometry of the targets.

## 6 Game Control and Scoring

Interactions between the user and the target shooting game can be modeled by a state diagram that shows all possible states and the set of rules that define the conditions for state transitions (see Figure 3).

After the *Initialization*, the game enters the *Ready* state where the gun is ready to fire and the trigger is checked against its threshold position for firing. The user must produce the commands to control the motion of the joints in the prosthetic arm (including shoulder, elbow, wrist, and index finger joints) to move the arm to a stable aiming posture and pull the trigger. Once the *Trigger Is Pulled*, the game enters the *Bullet Fired* state in which the bullet is fired and moves in its projectile trajectory and the collision between the bullet and the targets is checked. If a collision is detected with a target or if the bullet has passed the target area without any collision, the *Collisions Are Checked* and the game transitions to a *Handle Collisions* state. In this state, if there was a collision, it is handled, and the trigger is checked against its resting position. If the *Trigger Is Released*, the game enters the *Update Score* state where the score is updated. If the *Score Is Updated*, the game enters the *Ready* state, which completes a full firing cycle. The firing cycle can be repeated as many times as desired by the user. There are processes that are performed continuously irrespective of the game state, including the acquisition and decoding of the user commands, simulation of movement in response to control inputs and collision forces, calculation of fingertip forces for the haptic display, and

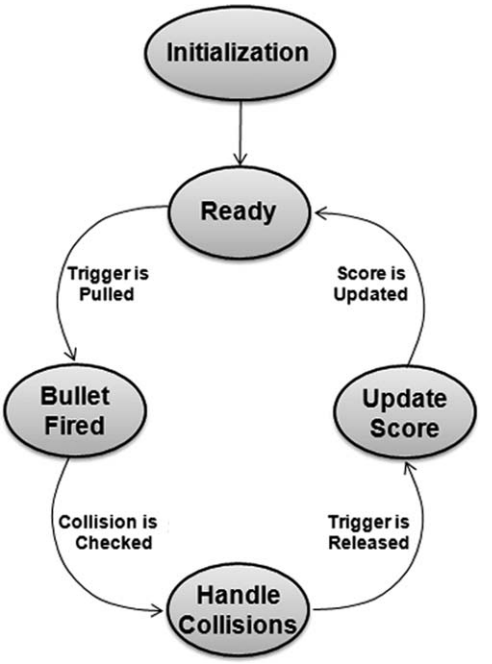


**Figure 3.** *Game control logic modeled by a state diagram. The state of the game can change only in the direction of the arrows subject to the rules of the state transition.*

display of the scene to the user, including the latest score and the appropriate sounds. The game control logic described above was modeled in Simulink using Stateflow toolbox (Mathworks, Inc.). The Stateflow toolbox is an extension to Simulink and is ideally suited for modeling the states and the event-driven state transitions in the target shooting game.

The points added to the total score depended on the difficulty of the targeting task. When the targets were missed, no points were added to the total score. The points for hitting the fixed circular target depended on the distance from its center. The points for hitting the moving circular target depended on its current speed of movement. The score for hitting the pendulum targets depended on their size and their current speed of movement. We intentionally kept the game logic simple because learning the skills to control a myoelectric prosthesis is usually difficult for the target amputee population (see Section 1). This allows the trainees to concentrate on learning the motivating and essential prosthetic control skills that directly affect their quality of life and well-being. As the trainee acquires more skills, the game's difficulty increases gradually to help maintain a high level of motivation and engagement.

## 7 Measurement of Execution and Rendering Times

We used the following methods to measure the execution time of the physics-based simulations in real-time PC and the rendering times of the virtual model in visualization PC.

Because the physics-based simulations must be executed in real time, they must be integrated using a fixed-step numerical integrator. Therefore, we had to identify a fixed-step numerical integrator and the largest integration time step we could use without losing simulation accuracy or causing instability. The largest allowable time step is important because the simulation code and any other real-time processes such as the acquisition of user commands must be executed within this time period. We first obtained a reference accurate simulation result by simulating the model with a variable step numerical integrator (Simulink's ode45). Starting from 0.001, we gradually reduced the numerical integrator's error tolerances until the simulation results stopped changing, indicating that the reference simulation is already accurate and its accuracy cannot be improved by further tightening of the error tolerances. We then simulated the same model with fixed-step explicit (Simulink's ode4) and implicit (Simulink's ode14x) numerical integrators and identified the largest fixed time steps that reproduced the reference simulation results. These were the largest allowable fixed time steps, because further increase in time step would make the simulation results inaccurate and unacceptable. Finally, the largest allowable time steps were compared to the actual execution times (in real-time simulation PC running xPC Target toolbox) to determine whether the model could be simulated in real time.

To measure the stereoscopic rendering times, all of the degrees of freedom in the model were animated by sinusoidal inputs to simulate a worst-case scenario. These motion data were generated by a Simulink model and sent via UDP at a much higher rate than MSMS can possibly render. This ensured that there were always animation data in the MSMS buffer for rendering. MSMS animated the motion data in the buffer as fast as it could, while counting the number of actually rendered data frames. The average rendering time per frame of motion data was found by dividing the total time of animation into the number of actually rendered frames in MSMS.

## 8 Results

The physics-based target shooting game has a significant amount of computation that must be performed in real time and at a rate that is adequate for closed-loop virtual training environments. To make these determinations, we evaluated the game in a test configuration in which various motion inputs (simulating the user's voluntary commands) were used to control the motion of the arm and the index finger to aim and shoot at the targets. We then evaluated the ability of the game controller to successfully manage the state transitions in response to the inputs, the ability of a typical simulation PC to simulate the physics-based movement of the limb and the game controller in real time, and the ability of a

typical visualization PC to render and visualize the virtual scene to the user in real time. The test configuration included a PC with Windows 7, 2 × 3.33 GHz Xeon CPU, 24 GB RAM, stereo-capable NVIDIA Quadro FX4800 graphics card, NVIDIA 3D Vision with shutter glasses, and a 3D stereo-capable ASUS 120 Hz monitor. The same PC was operated under Windows or real-time xPC Target operating systems to measure the rendering times and the simulation execution times, respectively.

The testing with various inputs showed that the game controller can control the state transitions smoothly and robustly. There was no interruption in the flow of the game as the arm was repeatedly moved to the aiming posture and the trigger was pulled. More importantly, the physics-based simulation of the movement and collisions provided a compelling sense of realism and immersion and appreciation of the interplay between the user actions and their consequences.

The largest allowable fixed integration time steps for the explicit and implicit integrators were 1 ms and 42 ms, respectively. The corresponding actual execution times per integration time step in the real-time PC were 0.58 ms and 5.50 ms. Clearly, the simulation model can be executed in real time using either implicit or explicit integrators. With the explicit integrator, the difference between the largest allowable time step and the actual execution time is only 0.42 ms (1 – 0.58 ms), which provides an inadequate margin for execution of additional but necessary real-time operations such as interactions with the user. But with the implicit integrator, there is an additional 36.5 ms (42 – 5.5 ms) that can be used as a safety margin to perform additional real-time operations. The average rendering time per frame of motion data in the visualization PC was 17.91 ms.

The total loop latency for the game includes the simulation execution time in real-time PC, the communication time for the UDP packets to travel from the real-time PC to the visualization PC, and the rendering time in the visualization PC. Because of the lack of specialized tools such as photodetectors, we did not measure the communication time for the UDP packets. But such measurements have been performed by other researchers who are using MSMS and its UDP communication protocol in their virtual training environment (Cunningham et al., 2011). According to their measurements (personal communication), the average UDP communication time is 3 ms. Therefore, the total loop latency for the explicit and implicit integrators were 21.49 ms (0.58 + 3 + 17.91 ms), and 26.41 ms (5.50 + 3 + 17.91 ms) resulting in real-time update rates of 46.5 Hz and 37.9 Hz, respectively.

## 9 Discussion

Learning the operation of an upper extremity neural prosthesis is difficult and stressful. This difficulty is expected to increase exponentially as prosthetic limbs acquire more degrees of freedom, and as more neural command channels from various sources are used to operate them. Training with physical prototypes is costly and potentially dangerous. Further, training with physical limbs is not readily available in the convenience of the home, and it is impractical in the recovery period after the amputation when training is critically important (Pons et al., 2005; Malone et al., 1984). To complement or replace the training with physical limbs, a number of virtual training applications have been developed in the past. To our knowledge, none has used physically realistic simulations of the limb and its interactions with the environment or made use of engaging and motivating games, probably because the development of such software has been prohibitively complex.

In this report, we have described the methods and a flexible software and hardware framework that can be used to develop a variety of high-fidelity simulations for virtual training and rehabilitation. The virtual models of the game were developed in MSMS software that is available free of charge. The simulation of movement, game control logic, and real-time simulations were implemented using Mathworks software toolkits that are widely available and are relatively inexpensive for academic users. Our expectation is that these readily available modeling tools will facilitate the development of physically realistic simulations and research studies that could elucidate the role and validity of physical realism in VR training applications.

We used a finite state diagram to model the game logic that smoothly controlled the state transitions in response

to the user inputs and object interactions. The overall loop latency for the game showed that the complex computations and rendering in the target shooting game can be simulated at real-time rates that are significantly higher than the minimum update rates of 20–25 Hz for closed-loop virtual training with visual feedback (Jorissen, Boeck, & Lamotte, 2006; Nichols, 1999). These update rates leave a large block of time for other application-specific processes such as the acquisition and processing of the neural command data. We are planning to obtain institutional ethics approval to clinically evaluate the utility of the physics-based target-shooting game as a safe and motivating environment to train the amputee users of the upper extremity prosthesis.

## Acknowledgments

The work described in this paper was funded by the DARPA Revolutionizing Prosthetics Program 2009. We thank Michelle Fung and Mehdi Khachani for their help with building MSMS models.

## References

Bouwsema, H., Sluis, C. K., & Bongers, R. M. (2010). Learning to control opening and closing a myoelectric hand. *Archives of Physical Medicine and Rehabilitation, 91*, 1442–1446.

Bridges, M., Beaty, J., Tenore, F., Para, M., Mashner, M., Aggarwal, V., et al. (2010). Revolutionizing prosthetics 2009: Dexterous control of an upper-limb neuroprosthesis. *Johns Hopkins Applied Physics Laboratory Technical Digest, 28*(3), 210–211.

Cunningham, J. P., Nuyujukian, P., Gilja, V., Chestek, C. A., Ryu, S. I., & Shenoy, K. V. (2011). A closed-loop human simulator for investigating the role of feedback-control in brain–machine interfaces. *Journal of Neurophysiology, 105*, 1932–1949.

Davoodi, R., & Andrews, B. J. (1998). Computer simulation of FES standing up in paraplegia: A self-adaptive fuzzy controller with reinforcement learning. *IEEE Transactions on Rehabilitation Engineering, 6*(2), 151–161.

Davoodi, R., & Loeb, G. E. (2011). MSMS software for VR simulations of neural prostheses and patient training and rehabilitation. *Studies in Health Technology and Informatics, 163*, 156–162.

Davoodi, R., Urata, C., Hauschild, M., Khachani, M., & Loeb, G. E. (2007). Model-based development of neural prostheses for movement. *IEEE Transactions on Biomedical Engineering, 54*(11), 1909–1918.

Dupont, A. C., & Morin, E. L. (1994). A myoelectric control evaluation and trainer system. *IEEE Transactions on Rehabilitation Engineering, 2*(2), 100–107.

Durfee, W. K., Mariano, T. R., & Zahradnik, J. L. (1991). Simulator for evaluating shoulder motion as a command source for FES grasp restoration systems. *Archives of Physical Medicine and Rehabilitation, 72*(13), 1088–1094.

Eriksson, L., Sebelius, F., & Balkenius, C. (1998). Neural control of a virtual prosthesis. In L. Niklasson, M. Bodén, and T. Ziemke (Eds.), *Perspectives in Neural Computing: Proceedings of ICANN '98*. Berlin: Springer-Verlag.

Fraser, G. W., Chase, S. M., Whitford, A., & Schwartz, A. B. (2009). Control of a brain–computer interface without spike sorting. *Journal of Neural Engineering, 6*(5), 055004.

Hauschild, M., Davoodi, R., & Loeb, G. E. (2007). A virtual reality environment for designing and fitting neural prosthetic limbs. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 15*(1), 9–15.

Holden, M. K., Dyar, T. A., Schwamm, L., & Bizzi, E. (2011). Virtual-environment-based telerehabilitation in patients with stroke. *Presence: Teleoperators and Virtual Environments, 14*(2), 214–233.

Jack, D., Boian, R., Merians, A. S., Tremaine, M., Burdea, G. C., Adamovich, S. V., et al. (2001). Virtual reality-enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 9*(3), 308–318.

Jorissen, P., Boeck, J. D., & Lamotte, W. (2006). Bringing haptics and physical simulation together: Haptic travel through physical worlds. *Computer Animation and Virtual Worlds, 17*, 179–187.

Kim, K., Colgate, J. E., Santos-Munne, J. J., Makhlin, A., & Peshkin, M. A. (2010). On the design of miniature haptic devices for upper extremity prosthetics. *IEEE/ASME Transactions on Mechatronics, 15*, 27–39.

Kim, S. P., Simeral, J. D., Hochberg, L. R., Donoghue, J. P., & Black, M. J. (2008). Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of Neural Engineering, 5*(4), 455–476.

Kirsch, R. F., Acosta, A. M., van der Helm, F. C. T., Rotteveel, R. J. J., & Cash, L. A. (2001). Model-based development of

neuroprostheses for restoring proximal arm function. *Journal of Rehabilitation Research and Development, 38*(6), 619–626.

Kuhlen, T., Kraiss, K. F., & Steffan, R. (2000). How VR-based reach-to-grasp experiments can help to understand movement organization within the human brain. *Presence: Teleoperators and Virtual Environments, 9*(4), 350–359.

Kuiken, T. A., Li, G. L., Lock, B. A., Lipschutz, R. D., Miller, L. A., Stubblefield, K. A., et al. (2009). Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *Journal of the American Medical Association, 301*(6), 619–628.

Lourenco, C. B., Azeff, L., Sveistrup, H., & Levin, M. F. (2008). Effect of environment on motivation and sense of presence in healthy subjects performing reaching tasks. In *Proceedings of Virtual Rehabilitation 2008* (pp. 93–98).

Malone, J. M., Fleming, L. L., Roberson, J., Whitesides, T. E., Leal, J. M., Poole, J. U., et al. (1984). Immediate, early, and late postsurgical management of upper limb amputation. *Journal of Rehabilitation Research and Development, 21*(1), 33–41.

Mulliken, G. H., Musallam, S., & Andersen, R. A. (2008). Decoding trajectories from posterior parietal cortex ensembles. *Journal of Neuroscience, 28*(48), 12913–12926.

Nichols, S. (1999). Physical ergonomics of virtual environment. *Applied Ergonomics, 30*, 79–90.

Nikooyan, A. A., & Zadpoor, A. A. (2009). Application of virtual environments to assessment of human motor learning during reaching movements. *Presence: Teleoperators and Virtual Environments, 18*(2), 112–124.

Nishikawa, D., Yu, W., Yokoi, H., & Kakazu, Y. (1999). EMG prosthetic hand controller using real-time learning method. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. I-153–I-158).

Nohama, P., Lopes, A. V., & Cliquet, J. A. (1995). Electrotactile stimulator for artificial proprioception. *Artificial Organs, 19*(3), 225–230.

Piron, L., Tonin, P., Piccione, F., Laia, V., Trivello, E., & Dam, M. (2005). Virtual environment training therapy for arm motor rehabilitation. *Presence: Teleoperators and Virtual Environments, 14*(6), 732–740.

Pons, J. L., Ceres, R., Rocon, E., Levin, S., Markovitz, I., Saro, B., et al. (2005). Virtual reality training and EMG control of the MANUS hand prosthesis. *Robotica, 23*, 311–317.

Riener, R., & Fuhr, T. (1998). Patient-driven control of FES-supported standing up: A simulation study. *IEEE Transactions on Rehabilitation Engineering, 6*(2), 113–124.

Rose, F. D., Attree, E. A., Brooks, B. M., Parslow, D. M., & Penn, P. R. (2000). Training in virtual environments: Transfer to real world tasks and equivalence to real task training. *Ergonomics, 43*(4), 494–511.

Scoufield, J. A., & Schreiner, S. (2004). Design of a myoelectric prosthetic training and assessment system. In *Proceedings of IEEE 30th Annual Northeast Bioengineering Conference* (pp. 200–201).

Sebelius, F., Eriksson, L., Balkenius, C., & Laurell, T. (2006). 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. *Journal of Medical Engineering and Technology, 30*(1), 2–10.

Smith, R. J., Huberdeau, D., Tenore, F., & Thakor, N. V. (2009). Real-time myoelectric decoding of individual finger movements for a virtual target task. In *Proceedings of IEEE Engineering in Medicine and Biology Society* (pp. 2376–2379).

Soares, A., Andrade, A., Lamounier, E., & Carrijo, R. (2003). The development of a virtual myoelectric prosthesis controlled by an EMG pattern recognition system based on neural networks. *Journal of Intelligent Information Systems, 21*(2), 127–141.

Van Verth, J. M., & Bishop, L. M. (2004). *Essential mathematics for games and interactive applications: A programmer's guide*. The Morgan Kaufmann Series in Interactive 3D Technologies. San Mateo, CA: Morgan Kaufmann.

Winter, D. A. (1990). *Biomechanics and motor control of human movement* (2nd ed.). New York: Wiley.