Video Article

Haptic/Graphic Rehabilitation: Integrating a Robot into a Virtual Environment Library and Applying it to Stroke Therapy

Ian Sharp[1], James Patton[1], Molly Listenberger[2], Emily Case[2]

[1]Department of Bioengineering, University of Illinois at Chicago and Rehabilitation Institute of Chicago

[2]Sensory Motor Performance Program, Rehabilitation Institute of Chicago

Correspondence to: James Patton at pattonj@uic.edu

URL: http://www.jove.com/video/3007
DOI: doi:10.3791/3007

Keywords: Bioengineering, Issue 54, robotics, haptics, virtual reality, wrapper class, rehabilitation robotics, neural engineering, H3DAPI, C++

Date Published: 8/8/2011

Citation: Sharp, I., Patton, J., Listenberger, M., Case, E. Haptic/Graphic Rehabilitation: Integrating a Robot into a Virtual Environment Library and Applying it to Stroke Therapy. *J. Vis. Exp.* (54), e3007, doi:10.3791/3007 (2011).

Abstract

Recent research that tests interactive devices for prolonged therapy practice has revealed new prospects for robotics combined with graphical and other forms of biofeedback. Previous human-robot interactive systems have required different software commands to be implemented for each robot leading to unnecessary developmental overhead time each time a new system becomes available. For example, when a haptic/graphic virtual reality environment has been coded for one specific robot to provide haptic feedback, that specific robot would not be able to be traded for another robot without recoding the program. However, recent efforts in the open source community have proposed a wrapper class approach that can elicit nearly identical responses regardless of the robot used. The result can lead researchers across the globe to perform similar experiments using shared code. Therefore modular "switching out"of one robot for another would not affect development time. In this paper, we outline the successful creation and implementation of a wrapper class for one robot into the open-source H3DAPI, which integrates the software commands most commonly used by all robots.

Video Link

The video component of this article can be found at http://www.jove.com/video/3007/

Protocol

Introduction

There is a growing need in all of human-machine interaction (HMI) for intuitive and efficient interactive environments. Numerous industries continue to depend more heavily on HMI, such as: rehabilitation robotics, the automotive industry, metals manufacturing, packaging machinery, pharmaceuticals, food, beverage, and utilities. Technologies employed in these industries include: display terminals, personal computers, and HMI software. These technologies may be combined together to perform unlimited functions.

Robots may be used to facilitate direct interaction with users, such as serving as a music instructor. For example, researchers at Waseda University have created a robot that plays the saxophone to teach people how to play and to understand the interaction between student and teacher [1]. Other robotics researchers have made a vision-based flying robot in order to determine how artificial intelligence may evolve into intelligent interactions with the environment [2]. The particular concentration of this paper resides within rehabilitation robotics.

Within the realm of research and industry, the quick pace of change for new products and user requirements continues to grow. These demands impose greater challenges in scalability. Therefore code design has become integral in meeting the needs of these entities in a timely manner. Hence, the quality of a strong architectural candidate would include easily interchangeable graphic-robot systems that include driver support. The H3DAPI architecture meets these needs and thus a wrapper class has been created. Furthermore, H3D is designed for virtual reality environments, such as those needed in rehabilitation robotics.

Neural rehabilitation robotics seeks to utilize robots for the purpose of assisting rehabilitation professionals. The assistance that these robots provide comes in the form of a force-field. Passed motor command researchers such as Shadmehr and Mussa-Ivaldi, used force-fields to foster motor adaptation, and have found 1) adaptation to an externally applied force field occurs with different classes of movements including but not limited to reaching movements, and 2) adaptation generalizes across different movements that visit the same regions of the external field[3]. Research from biomechanical engineers in Performance-Based Progressive Robot-Assisted therapy shows that repetitive, task-specific, goal-directed, robot-assisted therapy is effective in reducing motor impairments in the affected arm after stroke[4], but the exact therapeutic effects and parameters continue to be a field of research.

Sensory feedback affects learning and adaptation. Therefore the next logical question would be to ask whether or not artificially increasing the magnitude of such feedback would promote faster or more complete learning/adaptation. Some researchers have found that applying greater sensory feedback forces or visual cues to enhance mistakes can provide an adequate neurological stimulus to promote higher levels

of adaptation/learning[5,6]. This is known as "error augmentation". This phenomenon may be due to the fact that once outcomes of a motor control action deviate from the ideal, our internal model self-adjusts according to the magnitude of error. Consequently, as our internal model approaches the external environment, error in the execution of a task decreases.

Research continues to support prolonged practice of functionally relevant activities for restoration of function, although many current health care policies limit the amount of time patients can spend time with therapists. The compelling question is whether these new applications of technology can go further than simply giving a higher dosage of the current state of care. Human-machine interaction studies have revealed new prospects in the areas of motor learning, and can in some cases offer added value to the therapeutic process. Specialized robotic devices combined with computer-displays can augment feedback of error in order to speed up, enhance, or trigger motor relearning. This paper will present a methodology of using a developed system for a clinical intervention as one such example of the application of this technology.

1. Establishing HAPI wrapper class for a robot

1. Create a wrapper for HAPI the haptics library by creating your own .cpp and header file. For example we will use the name HAPIWAM.cpp and HAPIWAM.h.
2. Place HAPIWAM.cpp into the source directory: HAPI/src
3. Place HAPIWAM.h into the header file directory: HAPI/include/HAPI
4. At the top of HAPIWAM.h, include the main header file(s) of your robot, in the case of the Barrett WAM, that would be:

```
extern "C" {
#include <include/btwam.h>
}
#include <HAPI/HAPIHapticsDevice.h>
```

Note: extern "C" is required to resolve compiler mangling, because the included library is written in 'C' and the H3DAPI is written in C++.

5. In HAPIWAM.h, create your class and include the following 4 functions

```
bool initHapticsDevice(int);
bool releaseHapticsDevice();
void updateDeviceValues(DeviceValues &dv, HAPITime dt);
void sendOutput(HAPIHapticsDevice::DeviceOutput &d, HAPITime t);
```

6. Make sure your class inherits publicly from the HAPIhapticsdevice class.
7. Create a header guard for your class.
8. Create static DeviceOutput and static HapticsDeviceRegistration attributes under the HAPIWAM class.
9. Create your static member functions for callbacks.
10. Define your constructor and destructor in HAPIWAM.cpp.
11. Register your device in HAPIWAM.cpp.
12. Define your 4 inherited functions and callbacks in HAPIWAM.cpp.

2. HAPI library creation

1. Now that we have created the HAPI wrapper class, we need to build your wrapper into the HAPI library. The WAM depends on some libraries that H3DAPI does not depend on in its raw form, therefore these libraries will need to be added to HAPI. Go to HAPI/HAPI/build, and edit CMakeLists.txt. Add the dependent libraries after the line that says 'SET(OptionalLibs)'.
2. Open a command console and navigate to: HAPI/HAPI/build and type the following 3 commands in this order:

```
cmake .
sudo make
sudo make install
```

3. H3D wrapper class

1. To create the wrapper class for the H3D library with your HAPIWAM, first create WAMDevice.cpp in the source directory: H3DAPI/src
2. Place WAMDevice.h into the header file directory: H3DAPI/include/H3D
3. WAMDevice.h should contain the standard header file for all H3DAPI devices, with the name replaced to whatever you want.
4. WAMDevice.cpp should contain the standard source for all H3DAPI devices, with the name replaced to whatever you want.
5. Now that the wrapper class has been created, rebuild the H3DAPI library. Do this by editing CMakeLists.txt in the same way that was performed in step 2.1, only under the directory: H3DAPI/build.
6. Rebuild the H3DAPI library under the directory H3DAPI/build

```
cmake .
sudo make
sudo make install
```

4. Finite state machine

1. Every targeted reaching program requires the creation of a finite state machine to control the experimental protocol or practice regime. Typical state machines include: Start of trial, Launch, Target Contact, and End of Trial. An example of the function of each state, and criteria to transfer between states is listed below.
2. The Start of Trial requires the allocation of a target. Targets locations may be set randomly for each trial or may be set from a file. The start of the trial ends once the user has launched toward the target above velocity threshold, typically 0.06 meters per second.
3. The Launch state occurs after the start of the trial. This state ends either once the user touches the target or stays inside of the target for a period of time. Once the target is touched, this enables the Target Contact state.
4. Target Contact occurs during the Launch state. It may end as soon as the target is touched or after the subject resides within the target for a specific period of time. Once this time has elapsed, the End of Trial state is enabled.
5. The End of Trial state should signal the data collection software to mark the data file, in whatever parsing the software developer has used, to delimit the end of the each trial. Unless the final trial has been completed, the end of the End of Trial state enables the Start of Trial state.

5. Application: rehabilitation of the stroke patient

1. The robotic interface was designed to involve therapist expertise while using the robot to enable something that could not otherwise be done. The technology enabled application (described in more detail below) of error augmentation, which magnified the errors perceived by the patient, which for several well-known reasons enhances the relearning process (fig 1).
2. A three-dimensional haptics/graphics system called the Virtual Reality Robotic and Optical Operations Machine (VRROOM). This system, presented previously[6], combined a projected stereo, head-tracked rendering on a semi-silvered mirror overlay display with a robotic system that recorded wrist position and generated a force vector (fig 2).
3. A cinema-quality digital projector (Christie Mirage 3000 DLP) displayed the images that spanned a five-foot-wide 1280x1024 pixel display, resulting in a 110° wide viewing angle. Infrared emitters synchronized separate left and right eye images through Liquid Crystal Display (LCD) shutter glasses (Stereographics, Inc). Ascension Flock of Birds magnetic elements tracked motion of the head so that the visual display was rendered with the appropriate head-centered perspective.
4. Upon qualifying for the study, each participant's functional ability was evaluated by a blind rater at the start and finish of each treatment paradigm with a one-week follow-up after each and an overall 45-day follow-up evaluation. Each evaluation consisted of a range of motion (ROM) assessment performed in the VRROOM as well as clinical measures including: the Box and Blocks Assessment, Wolf Motor Function Test (WMFT), Arm Motor Section of the Fugl-Meyer (AMFM), and Assessment of Simple Functional Reach (ASFR).
5. An exotendon glove with wrist splint was utilized to assist in neutral wrist and hand alignment. The center of the robot handle was attached to the forearm was placed posterior to the radiocarpal joint so that its forces acted at the wrist but allowed motion at the hand.
6. The patient's arm weight was lessened using a spring-powered Wilmington Robotic Exoskeleton (WREX) gravity-balanced orthosis. The patient's instructed goal was to chase a cursor presented in front of them moved via a tracking device in the hand of the therapist (therapist teleoperation).
7. Patients practice three days per week for approximately 40-60 minutes, with the patient, the therapist, and the robot working together in a trio. Subject and therapist sat side-by-side, and the subject was connected to the robot at the wrist.
8. Each session began with five minutes of passive range of motion exercises (PROM) with the therapist, followed by approximately ten minutes for situating the patient in the machine. The subject then completed six blocks of movement training lasting five minutes each with two-minute rest periods between each block.
9. During training, participants viewed two cursors on the stereo display. The treating therapist manipulated one cursor while the participant controlled the other. Patients were instructed to follow the exact path of the therapist's cursor as it moved throughout the workspace.
10. Error augmentation was provided both visually and by forces generated by the robot. When participants deviated from therapist's cursor, an instantaneous error vector e was established as the difference in position between the therapist's cursor and the participant's hand. Error was visually magnified by a factor of 1.5*e* (m) as part of the error augmentation. Additionally, an error augmenting force of 100*e* (N/m) was also applied, which was programmed to saturate at a maximum of 4 N for safety reasons.
11. Every other treatment block consisted of specific, standardized motions that were the same for each session. The other blocks allowed the therapist to customize training at specific areas of weakness based on therapist expertise and their observations. The treatment protocol included the practice of specific movements for all participants; including forward and side reaching, shoulder-elbow coupling, and diagonal reaching across the body.
12. While practicing, day-to-day median error was measured as one outcome of the practice. Special attention was given to blocks of standardized motions that were the same for each session. These were compared to previous days to determine if any incremental improvement could be observed on a day-to-day basis, which can be reported to the patient, the therapist, and the caregivers (fig 3).
13. Primary measures of outcome were measured weekly, 1 week after the end of treatment, and 45 days post to determine the retention of benefits. Key outcomes were the Fugl-Meyer motor ability score and our customized arm reach test that measured range of motion.

6. Representative Results:

When the protocol is done correctly, then once the <AnyDevice> node is loaded into the H3DViewer or H3DLoad, the WAM device should be recognized and initiated. If the WAM were replaced with another robot, the code itself would not need to be changed.

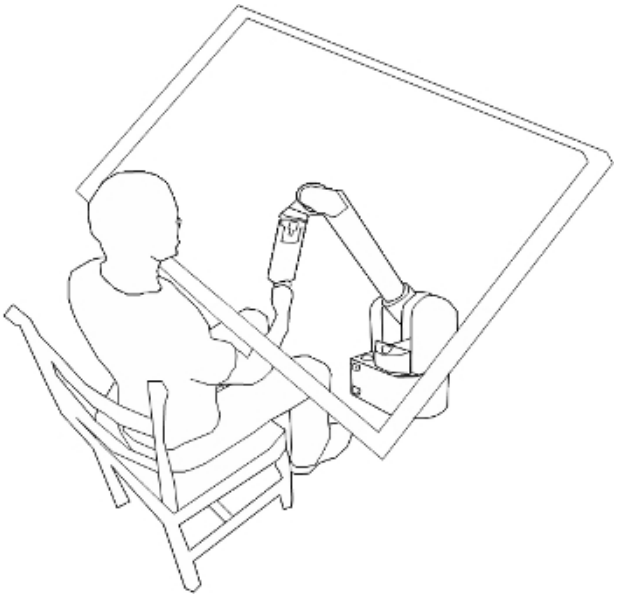

Figure 1. Subject seated at the haptic/graphic apparatus.

Figure 2. Subject seated at the haptic/graphic apparatus with physical therapist.

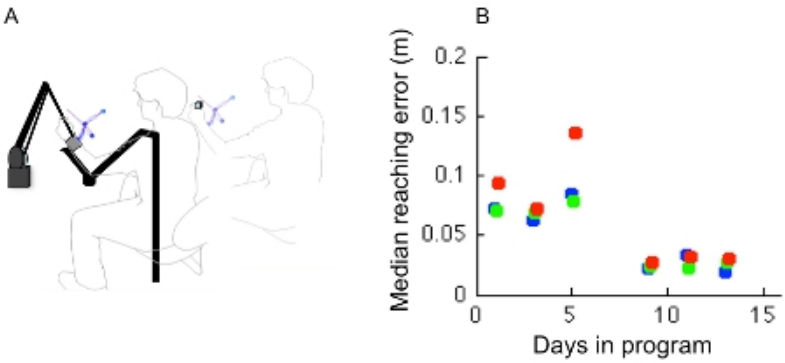

Figure 3. Configuration for rehabilitation of the stroke patient. A) subject and therapist working together, seated and using the large-workspace haptic/graphic display to practice movement. The Therapist provides a cue for the subject, and can tailor conditioning to the needs of the patient. The robot provides forces that push the limb away from the target and the visual feedback system enhances the error of the cursor. B) Typical chronic stroke patient improvement from day to day. Each dot represents the median error measured for a 2-minute block of stereotypical functional movement. While the patient shows progress across the 2-week period and overall benefit, this person did not always improve each day.

Discussion

This method of wrapper class implementation allows for different robots to be used, without changing the source code, when using the H3DAPI. Specifically, researchers who have written their haptic/graphic environment in H3D and tested their experiment with a phantom robot would be able to carry out the same or similar experiment using the Barrett WAM, and vice versa. This type of device independent cross-communication

carries implications for international rehabilitation robotics research. Such implications facilitate rapid haptic/graphic development, international research collaboration, and inter-research lab communication.

Rehabilitation robotics has yet to uncover the numerous parameters involved in motor learning. One of the time consuming steps during haptic/graphics development includes compilation time. With numerous rehabilitation parameters, compounded with the compilation time for each program, the development life cycle to test all possible group permutations rises rapidly. H3D, with its absence of compilation requirements, allows for quick development of numerous virtual reality scenes. This comes as an advantage for those researchers aspiring to probe the effects of various training scenarios.

Limitations of this 'hard-coded' wrapper class integration approach include the fact that this procedure must be repeated each time there is a new distribution of the H3DAPI. Possible modifications to integrating the wrapper class into your latest distribution of the H3DAPI would be to create the wrapper class separately from the H3DAPI. You would then put your wrapper class into a *.so library file. This would isolate your class from the original H3DAPI distribution.

Disclosures

The wrapper classes in this tutorial are under copyright by Ian Sharp.

Acknowledgements

I would like to acknowledge the technical help of Brian Zenowich, Daniel Evestedt and Winsean Lin.

References

1. Solis, J., Takeshi, N., Petersen, K., Takeuchi, M., & Takanishi, A. Development of the anthropomorphic saxophonist robot WAS-1: Mechanical design of the simulated organs and implementation of air pressure. *Advanced Robotics Journal*. **24**, 629-650 (2010).
2. Zufferey, J.C., Floreano, D. *Evolving Won-Based Flying Robots*. Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision, LNCS 2525, Berlin, Springer-Verlag, 592-600 (2002).
3. Conditt, M.A., Gandolfo, F., & Mussa-Ivaldi, F.A. The motor system does not learn the dynamics of the arm by rote memorization of past experience. *Journal of Neurophysiology*. **78** (1), 554 (1997).
4. HI Krebs, JJ Palazzolo, L. Dipietro, M. Ferraro, J. Krol, K. Rannekleiv, BT Volpe, and N. Hogan. Rehabilitation robotics: Performance-based progressive robot-assisted therapy. *Autonomous Robots*. **15** (1), 7-20 (2003).
5. K. Wei & K. Kording. Relevance of error: what drives motor adaptation? *Journal of neurophysiology*. **101** (2), 655 (2009).
6. Y. Wei, P. Bajaj, R. Scheidt, & Patton., J. Visual error augmentation for enhancing motor learning and rehabilitative relearning. In *IEEE International Conference on Rehabilitation Robotics*. 505-510 (2005).