Towards Software-Enabled Rehabilitation

Todd D. Murphey and Brenna D. Argall

Abstract— **Software-enabled rehabilitation techniques hold the promise of revolutionizing continuous, at-home, and remote care, both on-going and immediately post-trauma. However, critical challenges arise when automating the schedule of therapy and when automating tasks. Challenges include the software needs for dynamic tasks versus those of static tasks, human-in-the-loop control/safety, and interface design. This note discusses some of these challenges and how they relate to creating simple, inexpensive devices that can be used in various rehabilitation settings. In particular, software design plays a critical role in scaling down the complexity of some types of therapy.**

I. Introduction

Robotics has played a substantial role in physical therapy and physical augmentation in the last few years. Well designed robotic systems can physically support people during motion [4], keeping them safe and helping them avoid unsafe exertion. Such devices can also play a large role in the design of therapies. But software design rarely plays a central role in the creation of these robots; instead, most of the effort goes into make a robot that is naturally mechanically safe.

There are increasingly reasons to view software as one of the fundamental challenges in automated rehabilitation systems. First, software—once written—is essentially free and can be distributed across large populations easily. Insofar as any therapy is possible without a device supporting it, that means that therapy infrastructure can be provided for free. That doesn't mean that software can replace mechanically supported therapy approaches, but it does mean that there may well be types of therapy that can be accomplished without any use of physical infrastructure. Moreover, these software-enabled therapy tools might be used in conjunction with physical infrastructure like therapy machines, to enhance their operation.

This note discusses different ways in which therapy practices are, or can be, influenced by software design. It also focuses on what needs software could provide and how those needs push on engineering practice and how they require fundamental advances in algorithms. This paper focuses on four obstacles we have identified in the past few years of work in this area: 1) the difference between dynamic tasks and quasi-static tasks, 2) human-in-the-loop control, 3) interface design, and 4) implications for numerical methods used in automated therapy.

{t-murphey,brenna.argall}@u.northwestern.edu
Departments of Mechanical Engineering and Electrical Engineering & Computer Science, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA 60208

This material is based upon work supported by the National Science Foundation under award CNS—1329891 "CPS: Synergy: Collaborative Research: Mutually Stabilized Correction in Physical Demonstration". Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

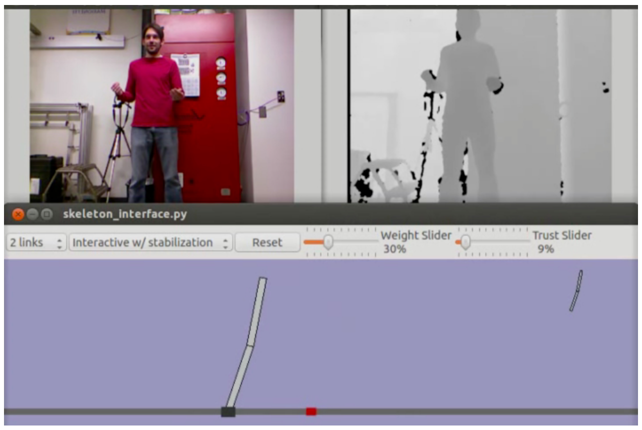
II. Software Support for Dynamic Tasks

Tasks in therapy involve motion. But whether that motion is sensitive to the particular decisions made in software (e.g., simulation) depends in many respects on how dynamic the tasks and motions are. Many of the tasks used for physical therapy are dynamic tasks for which destabilization is a risk; for example walking on a treadmill or balancing on balance board. Accordingly, a number of virtual reality therapy systems have incorporated dynamic tasks, like balancing and walking, into their gaming environments. There are simulation tasks, like virtually pushing a block on a table using quasi-static assumptions, that pose little risk of destabilization and creating a confusing environment for a subject to operate within. However, dynamic tasks—such as the one pictured in Fig. 1—can easily go unstable, either because the subject moved in a way that caused instability or because the numerical representation of the dynamics created instability. Clearly that latter is undesirable, but even the former creates problems in therapy because of intrinsic frustration in tasks that are—for the subject—impossible.

Control-aware software—software that either has task controllers available or that can synthesize controllers in real-time—can address subject frustration by providing the subject with controlled task support. This can come in the form of creating artificial impedance, or could potentially treat the user as a disturbance, or could provide a control signal that augments the subject control to maintain stability. Whatever the choice, software is playing a critical role in determining the usability of a therapy approach. In Fig. 1, we use a controller to *augment* a user's controller to ensure that task goals (e.g., stability of the pendulum in its inverted position) are achieved. Crucially, we also parameterize the amount of help the controller provides so that the subject has to be involved in the task.

III. Human-In-The-Loop Control

The previous section argues that control-aware software will be a critical component in the automated support of therapy. One key aspect of this support is that the subject will always be in the loop, participating in control. Moreover, the subject may not behave predictably—and the statistics of motion can be expected to potentially be far out on the "tails" of human motion—but the subject's motion needs to always participate actively in motion determination (e.g., motion direction/speed), motion stability (e.g., insisting on


skeleton_interface.py

2 links | Interactive w/ stabilization | Reset | Weight Slider 30% | Trust Slider 9%

Fig. 1. An example of software-supported motion direction. A person (upper left) demonstrates motion using the Kinect sensor (upper right) to control a virtual double inverted pendulum (bottom). The person's motion is augmented by the synthesized feedback controller so that the double inverted pendulum motion projects back to a stable, feasible motion.

eigenvalues of a linear dynamical system be in the left of the complex plane), and motion safety (e.g., not allowing trajectories to enter unsafe/undesirable regions). The question then arises: For what classes of motion can we actually do this analysis, and for what classes of motion can we synthesize controllers in real-time, and for what classes of motion can we analyze the synthesized controllers in real-time? All three of these questions need to be addressed for any task with which one intends to engage a subject.

A. *The Role of Receding Horizon Control in Therapy*

The main issue with augmenting human motion with embedded controllers is that control synthesis is challenging, both analytically and computationally, and "closing the loop" improperly can often make a system *more* unstable that it was originally. Four factors play dominant roles in control design for subject-in-the-loop control. First, the body and the environment it operates in are relatively high-dimensional. At a minimum, the body needs to be regarded as having 20-40 degrees of freedom, and the environment can be equally complex. Second, the body and its environment are often nonlinear. Third, nonsmooth behavior can play a substantial role in the dynamics, particularly during the intermittent contact experienced during locomotion. Lastly, even if a controller has been synthesized for all these needs, verification of the controller (e.g., identifying its basin of attraction and guaranteeing trajectories cannot enter unsafe states) is needed. These four needs are challenging even in the most benign settings (e.g., low dimensional, linear systems with unilateral constraints and safety sets described by hyperplanes in the state space). There are two natural conclusions to reach. First, we should only assign tasks for which these problems are solvable when working with subjects. Second, we need new software tools that extend these capabilities to more relevant tasks of high(er) dimension (e.g., [2]), nonlinearities (e.g., again [5]), non smooth behavior [3], and verification.

B. *Compressing High Dimensional States in an Interface*

The first factor in control design mentioned above—the high dimensional nature of the control problem—is not only a problem for control synthesis. It is also a serious challenge for the subject, particularly when we provide the subject with an interface intended to assist them in the determination of a policy for task completion. If we provide feedback to the subject, that feedback has to in some manner compress high dimensional tasks into lower dimensional, understandable settings. For instance, in [6] we used linear quadratic methods to provide vibrotactile feedback to the user in Fig. 1. Linear quadratic control laws can be thought of as a type of compression—they take the state in $\mathbb{R}^n$ and return a signal in the control space $\mathbb{R}^m$; it is common for $m << n$. Regardless of how compression occurs, it is vital to compress state information in any interface that provides feedback to a subject.

IV. Choosing Numerical Methods

A minor note on numerical methods is warranted. We use `trep`, available at *http://trep.googlecode.com*, developed in our lab [1] so that we have numerical routines that are capable of i) simulation, ii) estimation and iii) linear and nonlinear control synthesis, all in an internally consistent setting. Certainly the variational integrators that `trep` uses are not the only reasonable choice; there are likely many choices one could make and still have these three needs be coherent with each other. However, we have benefitted substantially from this choice, particularly when it comes to bandwidth and sensitivity to delays. Controllers for operator-in-the-loop control of a swinging mass system [5] can run experimentally at very low frequencies, including frequencies *lower* than the Nyquist frequency for the linearized equations. The same control system, changing only the numerical method to Euler integration, does not perform acceptably at frequencies an order of magnitude greater. This is only to say that, as we design embedded systems that support human motion, it may well be important to choose numerical methods carefully in order to have software that is reliable.

V. Conclusions

Software will play an increasingly important role in automating therapy processes. The developed software will need to have many characteristics, including extraordinary reliability in the presence of substantial environmental challenges. Human-in-the-loop control presents many challenges, and human-in-the-loop control with subjects requiring therapy simply has amplified versions of these challenges. Specifically, dynamic tasks—such as balance during walking—will require special attention.

References

[1] E. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, 25(6):1249–1261, 2009.

[2] T. D. Murphey and B. Argall. Making robotic marionettes perform. In *ICRA Workshop on Robotics and Performance Arts: Reciprocal Influences*, 2012.

[3] D. Pekarek and T. D. Murphey. A projected Lagrange-d'Alembert principle for forced nonsmooth mechanics and optimal control. In *IEEE Int. Conf. on Decision and Control (CDC)*, pages 7777 – 7784, 2013.

[4] Michael Peshkin, David A Brown, Julio J Santos-Munné, Alex Makhlin, Ela Lewis, J Edward Colgate, James Patton, and Doug Schwandt. Kineassist: A robotic overground gait and balance training device. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pages 241–246. IEEE, 2005.

[5] J. Schultz and T. D. Murphey. Extending filter performance through structured integration. In *American Controls Conf. (ACC)*, 2014.

[6] E. Tzorakoleftherakis, F. Mussa-Ivaldi, R. Scheidt, and T. D. Murphey. Effects of optimal tactile feedback in balancing tasks: a pilot study. In *American Controls Conf. (ACC)*, 2014.