Making Robotic Marionettes Perform

Todd D. Murphey and Brenna Argall

***Abstract*— This paper describes a project with the long term goal of automated performance marionettes, accomplished by capturing human motion and automating the motion imitation synthesis for an experimental marionette system. The automation and performance goals required the development of hardware and software tools that enable motion imitation, leading to a series of results in numerical simulation, optimal control, and embedded systems. Marionettes are actuated by strings, so the mechanical description of the marionettes either creates a multi-scale or degenerate system—making simulation of the constrained dynamics challenging. Moreover, the marionettes have 40-50 degrees of freedom with closed kinematic chains. Choreography requires the use of motion primitives, typically originating from human motions that one wants the marionette to imitate, and resulting in a high dimensional nonlinear optimal control problem that needs to be solved for each primitive. Once acquired, the motion primitives must be pieced together in a way that preserves stability, resulting in an optimal timing control problem. We conclude with our current results that enable the synthesis of optimal imitation trajectories, and overview the next steps we are taking in this project towards automated performance marionettes.**

I. INTRODUCTION

This paper provides an overview of, and future directions for, a project focusing on robotic marionettes. The project (now five years old) was heavily motivated by the example of puppet choreography, where motions are split into simple primitives and concatenated in time using counts that help the puppeteers synchronize their motions (e.g. Fig.1). This example suggests a formal approach to motion synthesis for complex systems that is very appealing—to first construct primitive motions, potentially as a batch computation prior to execution, and then piece those primitives together in a way that ensures stability. Moreover, the marionettes are severely underactuated, subject to unilateral constraints (because of the strings being able to go slack), mechanically degenerate (because of the strings being so light), and their articulated rigid body dynamics are nonlinear. Marionettes therefore seemed a reasonably challenging system to consider: it was clear that synthesizing stabilizing feedback controllers for the marionettes would be challenging. Success however also seemed likely: if puppeteers could do it, it seemed plausible to be possible computationally as well.

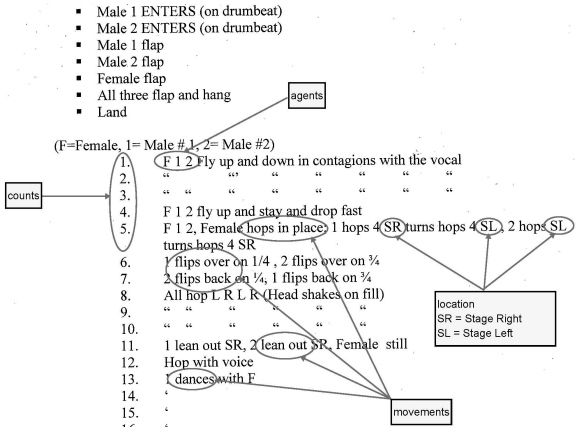


Fig. 1. An example of choreography from the Atlanta Center for Puppetry Arts [9]. Choreography includes basic primitive motions that are concatenated together in time. When multiple puppets are on stage, counts are used to facilitate synchronization.

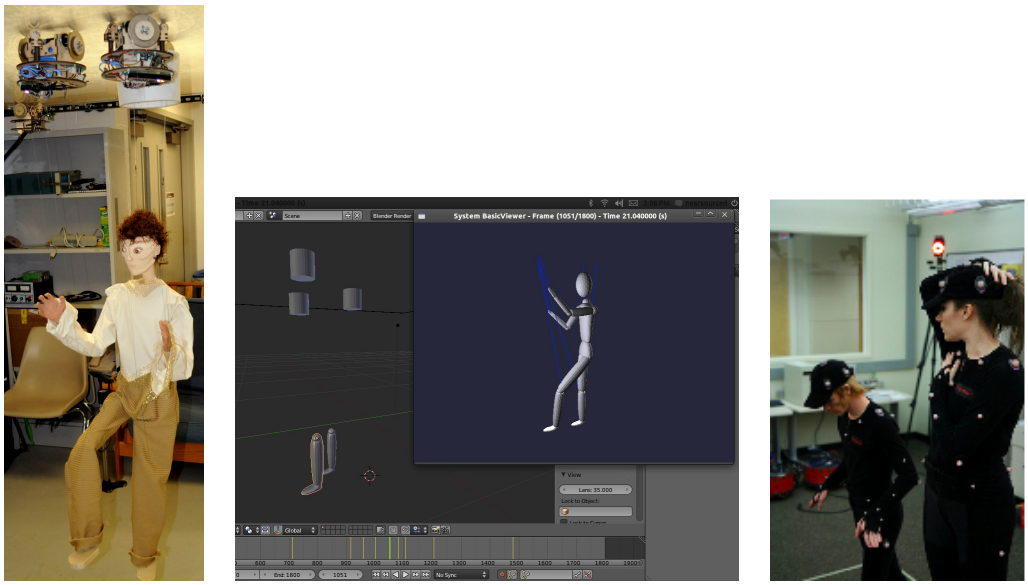
Our plan was to first decide on an experimental approach, and then choose a numerically feasible manner to simulate the marionettes and compute control policies. What actually happened was more complex than that; the experimental approach and software approach have been iterating with each other for several years. Indeed, many of the calculations we are accustomed to performing for simpler systems were numerically ill-posed for the marionettes. *Moreover, one of the main things we have learned thus far in this project is that artistic performance creates a constraint on the project outcome, which in turn requires flexibility in the various technology design stages.* Thus, the experimental system had nontraditonal requirements—that is, artistic performance requirements—regarding the types of inputs one should use to drive the system. For instance, controllers that heavily damped motion in order to stabilize it are unacceptable from a performance perspective because the motion is not visible to the audience.

The challenges that we encountered can roughly be placed in three categories. First, simulation of the marionettes was much more difficult than anticipated. Second, feedback control synthesis was nontrivial, and even the form of inputs to the system was not clear. (E.g., should marionette strings be modeled as rigid or as unilateral constraints? Puppeteers

T.D. Murphey t-murphey@u.northwestern.edu
B. Argall brenna.argall@northwestern.edu
Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA 60208
Electrical Engineering and Computer Science and Physical Medicine and Rehabilitation, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA 60208

This material is based upon work supported by the National Science Foundation under award IIS-0917837. 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.

This work has been a collaborative effort of many individuals, including Prof. Magnus Egerstedt at Georgia Tech, Dr. Lanny Smoot at Disney Imagineering, and several graduate students and undergraduate students at both Northwestern, Georgia Tech, and the University of Colorado. Anything in this paper represents the viewpoint of the authors only, and does not reflect the view of these other contributors.



(a) (b) (c)

Fig. 2. (a) The marionettes are suspended by differential drive vehicles that have winches to pull on strings; the marionettes have real-time state feedback at 30 Hz from a Kinect sensor. The differential drive vehicles are suspended by magnetic wheels and communicate wirelessly with a central computer running the Robot Operating System (ROS). (b) The `trep` software used to simulate and control the marionettes has visualization capabilities allowing the user to see if desired motions are stable in the simulated environment. (c) The goal is to take the motion capture data from dancers and automatically synthesize a motion program for marionettes to imitate the dancers' motion [16].

told us that if the strings ever go slack, they redesign the mechanical puppet, suggesting that treating the strings as constraints is reasonable.) Third, practical requirements such as control of the individual actuators (the differential drive vehicles in Fig.2(a)) was challenging because the actuators are all coupled to each other through the dynamics of the marionettes.

This paper is organized as follows. In Section II we overview the experimental system description. In Section III we provide our view of the "typical" approach to solving control problems for mechatronic systems; we discuss the various ways in which those approaches needed to be adapted to function in the present setting and how our software reflects those needs. In Section IV we discuss the current state of the project, with the upshot being that we can compute optimal imitation trajectories roughly only an order of magnitude slower than real-time. We end in Section V with a discussion of the next research direction that this work is taking—including learning by demonstration—and the importance of using the marionette performance as an end goal.

II. Experimental System Description

This work is a collaboration between Northwestern, Georgia Tech, the University of Colorado at Boulder, the Atlanta Center for Puppetry Arts, and Disney Imagineering. Disney Imagineering has played a central role in helping develop the hardware platform partially because animatronics in theme parks are very heavy, slow, and expensive and robotic marionettes promise to be both more agile and less costly and cumbersome. However, controlling marionettes is a very difficult technical problem; the marionettes have many degrees of freedom, have mechanical degeneracy due to the strings, and are highly constrained.

Why choose experimental marionettes? They have several key advantages. First, it is clear what they should *do*—because they are humanoid and are used to act out human motions, the goals of the mechanism are clear. Indeed, generating reference data for the marionettes can be as simple as using a Microsoft Kinect sensor to track one's own body. Second, it is clear that they *cannot* achieve those goals, at least not exactly. The dynamics of the marionette are radically different from those of a person, making exact motion replication impossible. Third, picking up a marionette, or watching a professional manipulate a marionette, tells us that close approximation of human motion should be possible, at least in many instances, and when it is impossible we can hope to *know* it is impossible. We know that puppeteers can solve these high dimensional motion planning problems—puppeteers do successfully get marionettes to convincingly imitate human motion—so we know the problems are solvable.

We have an animatronic marionette, pictured in Fig. 2(a), that has 40 degrees of freedom (22 dynamic degrees of freedom, 18 kinematic degrees of freedom, and 6 constraints due to the strings). We can simulate and linearize to both first and second order the marionette using the `trep` software developed in our laboratory over the last five years. Hence, we can compute feedback controllers to regulate trajectories, and can perform nonlinear control synthesis. (As an example, we can synthesize walking trajectories for the marionette directly from reference walking motion data.) The differential drive robots, used to winch and manipulate the

strings, are controlled using feedback controllers based on differential flatness properties. Then we compute optimal paths for the robots to follow, and string lengths for them to create, based on the puppets' dynamics and the desired motion. The techniques we use are briefly described in the next sections.

III. Typical Approaches to Control of Mechatronic Systems

Efficient methods in simulation for highly articulated rigid body systems have been studied for many years [2], [3], [4], [12], [30], [31]. However, less emphasis on *control* calculations for these same systems has been present (often times motivated implicitly by the use of probabilistic planners, which only use sampling of the dynamics to generate policies). In contrast, partially because of the high number of degrees of freedom and the highly dynamic behavior they exhibit, control of the marionettes benefits from numerical methods that provide both simulation and control simultaneously in order to keep computations tractable.

A. Dynamics

Dynamics are often represented in the form

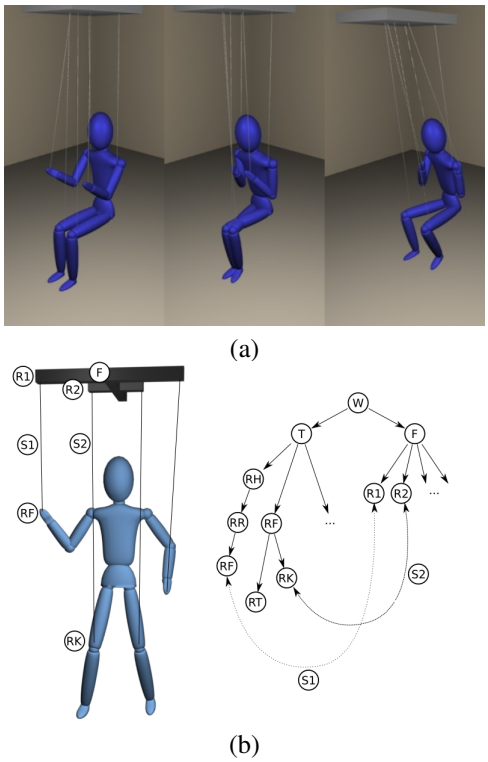
$$\dot{x} = f(x, u) \tag{1}$$

where $x = (q, \dot{q})$ and $q \in Q$ describes the configuration of the system. For rigid body systems, it has historically been convenient to write down the rigid body system in Newton-Euler coordinates; i.e., $Q = SE(3)^n$, where n is the number of rigid bodies in the system. This yields a state space of dimension $12n$ that is subject to constraints. For the marionette, for instance, just the body has 10 rigid bodies, so the state space for just the body would be 120 dimensions. If one includes the actuators, one adds a minimum of one degree of freedom for the length of each of the five strings and three to six degrees of freedom for each of the five actuators (depending on whether the actuators are planar or not). For the marionettes this brings the total nominal dimension of the state space up to $12{\cdot}10{+}2{\cdot}5{+}12{\cdot}5 = 190$. It should be clear that we don't want to be solving for feedback controllers in a 190 dimensional space if we can avoid it.

Because of these issues, we don't want to represent Eq. (1) as Newton-Euler equations and instead insist on working in generalized coordinates. In the case of the marionettes, this reduces the dimension of the state to $2m$, where m is the number of generalized coordinates. This give us 22 dynamic degrees of freedom for the marionette itself and another 18 degrees of freedom for the actuators, yielding 80 states. By utilizing a kinematic reduction [5], [6] we can reduce the state of the actuators down to 18 because the actuators are fully actuated. This gives us equations of motion

$$\begin{aligned} \dot{x}_a &= u \\ \dot{x}_p &= f(x_p, x_a) \end{aligned} \tag{2}$$

where x_a is the kinematic configuration of the actuators and $x_p = (q_p, \dot{q}_p)$ is the dynamic configuration *and* velocity of the marionette itself. (For details on this, see [21].) This



(a)

(b)

Fig. 3. Simulation of complex rigid bodies can take advantage of the mechanical topology of the system. For instance, a marionette is being simulated in (a) using a tree structure representation of the humanoid form in (b), with the constraints represented by cycles in the graph (see [20]).

leaves us with a much smaller, more manageable system to work with that only has a total of 62 dimensions in its state space. Nevertheless, 62 is still a comparatively large state space to be solving optimal control problems in, so the numerical techniques employed need to be as efficient as possible.

The last thing to point out is that because of the strings there are holonomic constraints relating the end of the arm to the length of the string (e.g., when the string length is constant L, the end of the arm evolves on a sphere of radius L). This creates a constraint

$$h(x_a, q_p) = 0 \tag{3}$$

that must be maintained during the simulation.

As previously mentioned, the continuous representation of dynamics found in Eq. (1) is not what we actually use to do computations. Moreover, when there are constraints, such as those seen in Eqs. (3), standard methods such as Runge-Kutta methods fail to preserve the constraints. Typically one would use solvers designed for Differential Algebraic Equations (DAEs) that project the numerical prediction onto the set of constrained solutions defined by the constraint in Eq. (3). We have found, however, that for high-index DAEs such as the marionette a tremendous amount of "artificial stabilization" is required to make the simulation of the DAE stable. This artificial stabilization—which typically takes the constraint $h(q)$ as a reference and introduces a feedback law that "stabilizes" the constraint—changes the dynamics

of the system, and if the feedback gain is high often creates a multi-scale simulation problem that is incompatible with real-time operation. As an alternative, we consider variational integrators [17], [22], [23], [24], [25], [29], [32].

Variational integration methods use the stationary action principle as a foundation for numerical integration that does not involve differential equations. This approach has several advantages–known conservation properties (such as guarantees about conservation of momenta, the Hamiltonian, and the constraints) as well as guaranteed convergence to the correct trajectory as the time step converges to zero. More importantly, variational integration techniques exactly simulate a *modified Lagrangian* system where the modified Lagrangian is a perturbation of the original Lagrangian. The Discrete Euler-Lagrange (DEL) equations are

$$D_1 L_d(q_k, q_{k+1}, k) + D_2 L_d(q_{k-1}, q_k, k) = F_k \quad (4)$$

$$h(q_{k+1}) = 0 \quad (5)$$

where L_d is a discretized form of the Lagrangian and F_k is an external force integrated over the k time step. This forms a root solving problem in which, given q_{k-1} and q_k, one solves for q_{k+1}. Repeating this rootsolving procedure forms the basis of simulation. Using this method, we can simulate the marionette considerably faster than real-time (using time steps of $0.01\ s$) without adding any sort of numerical heuristics such as artificial stabilization.

Let's say we start from the DEL equations and assume, by application of the implicit function theorem, that the solution exists and is locally unique [26]. Then, once we have made a choice of state (we choose $x_k = (q_k, p_k)$ where p_k is the generalized momentum), we have an update equation of the form

$$x_{k+1} = f_k(x_k, u_k)$$

just as we would if we had started from a differential equation. That is, the general form of the discrete time equation we wish to optimize is no different–in principle–in the variational case than it is in the standard ODE case. One can calculate an *exact* linearization of the DEL equations, including constraints and closed kinematic chains [18].

The approach taken in our work is based on the methods presented in [20] (based on [12]). Systems are represented as graphs where each node is a coordinate frame in the mechanical system and the nodes are connected by simple rigid body transformations (typically translations along and rotation about the X, Y, and Z axes, though any rigid body screw motion can be used). Transformations are either constant or parameterized by real-valued variables. The set of all variables establishes the generalized coordinates for the system. Figure 3 is an example of a marionette. The graph description can include closed kinematic chains, but in practice the graph is converted to an acyclic directed graph (i.e, a tree) and augmented with holonomic constraints to close the kinematic chains. This approach leads to fast ways to calculate $f(\cdot,\cdot)$ in Eq. (2) and $h(\cdot)$ in Eq. (3). Moreover, one can use the same structure to efficiently calculate the linearization [18], which is critical to nonlinear optimal control calculations, discussed next.

B. Nonlinear Optimal Control

Optimal control typically starts out with a cost function of some sort, often of the form

$$J = \int_{t_0}^{t_f} L(x(t), x_{\text{ref}}(t), u(t))dt + m(x(t_f), x_{\text{ref}}(t_f)) \quad (6)$$

where $L(\cdot)$ represents a weighted estimate of the error between the state and the reference state (which is potentially not a feasible trajectory for the system), and $m(\cdot)$ a the terminal cost at $t = T$. The $x_{\text{ref}}(t)$ is the reference data we wish the marionettes to imitate. Minimizing this cost function subject to the dynamics in Eqs. (2) and (3) can be accomplished using iterative descent methods. In particular, one uses the equivalence between the constrained minimization and the unconstrained minimization of the objective function composed with a differentiable projection $\mathscr{P}(\cdot)$ onto the constrained subspace. That is, the two minimizations

$$\min_{v \in W \subseteq V} g(v) = \min_{v \in V} g(\mathscr{P}(v))$$

(where V is the vector space and W is the differentiable submanifold of admissible vectors) are equivalent [15]. The projection operator $\mathscr{P}(\cdot)$ comes from computing a feedback law. In particular, if one interprets the "gradient" descent algorithm as starting at some nominal trajectory $\xi = (x(t), u(t))$ and solving for a descent direction $\zeta = (z, v)$ that optimizes the local quadratic model

$$\zeta = \arg\min_{\zeta} Dg(\xi) \cdot D\mathscr{P}(\xi) \cdot \zeta + \|\zeta\|^2,$$

then one just has to solve a standard time-varying LQR problem. This means that one has to be able to compute the time-varying linearization

$$\dot{z} = A(t)z + B(t)v \quad (7)$$

where $A(t) = \frac{\partial f}{dx}(x(t), u(t)) = D_1 f(x(t), u(t))$ and $B(t) = \frac{\partial f}{du}(x(t), u(t)) = D_2 f(x(t), u(t))$. One has to be able to do so for arbitrary trajectories in the state space, potentially including infeasible trajectories (in the case of linearizing about the desired trajectory). Solving for both the descent direction *and* the projection operator itself involves solving the Riccati equations

$$\dot{P} + A(t)^T P + PA(t) + Q - PB(t)R^{-1}B(t)^T P = 0 \quad (8)$$

at each iteration, where $Q > 0$ and $R \geq 0$ are time-varying local estimates of the cost of the state and control respectively. We follow this outlined procedure, but adapt each step to the discrete variational case, so that the dynamics are described by the DEL equations, the linearization is the discrete time linearization of the DEL equations (instead of the infinitesimal linearization), and the feedback law for the projection operator comes from solving a discrete time Riccati equation. This is surprisingly efficient; as discussed in Section IV, solving for a $10\ s$ walking motion takes a few minutes even for a 62 state nonlinear system.

C. Choreography and Hybrid Optimal Control

In [9], [27], [28] we developed an optimal control interpretation of choreography. In particular, we formalize choreography as a sequence of *modes* that can be pieced together to form a script. Each mode has its own dynamics, creating a system with dynamics

$$\dot{x} = f(x(t), u(t)) = f_i(x(t), u(t)) \quad t \in (\tau_i, \tau_{i+1})$$

where each i corresponds to a different mode of the system. To optimize such a system, one needs to be able to minimize an objective function J with respect to the switching times τ_i of the system. This derivative $\frac{\partial J}{\partial \tau_i}$ with respect to the switching times depends on the switching time adjoint equation

$$\dot{\rho} + A(t)^T \rho + \frac{\partial L}{\partial x} = 0 \tag{9}$$

along with a boundary condition at $\rho(t_f)$ (see [7], [10], [11], [19]). This adjoint equation only needs to be computed once to compute all the derivatives of J.

It is tempting to simply create the discrete time analog of this hybrid timing control problem, just as we did in Section III-B, but in discrete time this timing control problem becomes ill-posed. In particular, it is nonsmooth, for all order of both explicit and implicit numerical methods [13] (but appears to be provably locally convex if and only if the continuous time optimization is locally convex). Hence, although we have not successfully solved the timing control problem yet, the marionette example forced us to carefully investigate what happens in discrete time, leading to surprising results.

IV. Optimal Motion Imitation for Marionettes

So where is the project now? We can do optimal control for a full dynamic model of a marionette, such as that seen in Fig. 2(b). Using a 10 s walking motion that involves the full body dynamics and the actuators, `trep` (available at *http://trep.googlecode.com*) can compute a locally optimal trajectory in about five minutes (depending on processor speed and memory). (It can compute a "reasonable" trajectory much faster than that.) The robotic marionettes will be on public display for the first time on April 14, 2012 at the Chicago Museum of Science and Industry.[1]

V. Towards Physical Learning By Demonstration

The next steps of this project focus on how to program the marionettes for performance. A person may have very concrete ideas about *what* an autonomous, embedded system is supposed to accomplish without having very good ideas about *how* the system can be expected to accomplish it. Moreover, that person may not be able to translate goals for a system into a mathematical context understandable by a machine, leading to the breadth of research in *Learning from Demonstration (LfD)* [1], where the system infers in mathematical terms the goals implicit in the act of demonstration. Lastly, the system itself rarely has the ability to generate goals for itself, but it may have very technically precise information about what its own capabilities are (e.g., local properties such as controllability and observability are computable). Balancing the ability of a person to direct a system—typically in terms of infeasible system evolutions—against the system's representation of its own capabilities and limitations is the purpose of our next research direction within the marionette project.

The goal is for a human operator of the system to be able to physically manipulate the marionette and thereby manipulate its control system, all while receiving feedback from the system about the feasibility of the operator-provided trajectories and without ever interacting with a computer keyboard (thereby providing an intuitive interface by which a human instructor is able to impart artistic performance goals). To date, LfD has been a largely data-driven technique, with neither the initial derivation nor the subsequent adaptation of the resulting control behaviors being verified for feasibility or stability. Moreover, if one has a system where controller stability has implications for system safety, policy adaptation becomes significantly nontrivial. By contrast, sophisticated stability analysis is possible for control behaviors derived via optimal control and other control design methodologies; however, to date such formulations are rarely amenable to human instruction. Moreover, until recently the formal verification of nonlinear control strategies has been largely limited to low-dimensional systems where the equations of motion and their derivatives can be computed by hand. (This is essentially the gap filled that `trep` and a few other software packages (e.g., Dymola)). We posit that for accessible and intuitive instruction of complex systems, characteristics of both approaches are crucial: that is, the development and adaptation of control behaviors by operators who are not control or robotics experts, and the ability to verify and thus ensure the stability of those control behaviors.

We therefore are taking an approach that is a synergy between these two areas: that combines control theoretic calculations and analysis with demonstration-based behavior adaptation. In particular, the proposed approach first derives an initial control behavior via optimal control, then engages a human teacher to provide physical guidance for corrective demonstration, and finally verifies that the controller produced as a result of the inferred corrections is in fact stable. Control theoretic techniques furthermore are used to engage the teacher, by having the software identify regions of difficultly for the nonlinear optimal control algorithms and solicit the teacher to provide instruction that overcomes the instability.[2] The teacher also might provide instruction at her own discretion, as traditionally happens in LfD paradigms.

The experimental target is that by the end of this portion of the project, an operator will be able to physically manipulate the marionette to produce a desired motion—embedded

[1] So by the time of this workshop we will be able to provide video of the marionettes in a performance of sorts.

[2] While active teacher engagement by the learner is not a new idea in LfD, the focus has not been on the explicit analysis of controller stability—rather, for example, on addressing dataset sparsity or ambiguity [8], [14].

within which might be artistic performance goals—all while getting feedback from the automatic control system about the stability of that motion. A key point is that the operator will *only* interact with the physical system, rather than interacting with software. During the interaction, both the software *and* the person must be sufficiently aware of the net dynamics to ensure stability while modifying motion. Hence, the operator will be able to physically manipulate the marionette to achieve desired goals, altering its dynamics enough to help it make progress without altering them so much so as to make the motion only feasible with help.

VI. CONCLUSIONS

Controlling marionettes is more complicated than it may seem. It requires reliable simulation techniques capable of handling degeneracies and closed kinematic chains in a reliable manner. It requires control that is specifically tailored to those numerical methods, and it requires a software architecture that is sufficiently flexible to incorporate experimental changes as they occur. Even with motion imitation now effectively a solved problem, concatenating motions is still nontrivial since the dynamics are sensitive to the transitions between motions. Hence, this project is still a good ways off from successfully making a robotic marionette perform a fully choreographed story.

As discussed in Section V, our next steps include the incorporation of LfD techniques into a feedback control framework. Key characteristics will include the ability for a human to physically instruct the robotic platform as the system reasons explicitly about control stability, with the end result of an intuitive and computationally stable interface for imparting artistic targets, within the larger goal of automated marionette performances.

REFERENCES

[1] Brenna Argall, Sonia Chernova, Brett Browning, and Manuela Veloso. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5), 2009.

[2] D. Baraff. Non-penetrating rigid body simulation. In *State of the Art Reports*, 1993.

[3] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH*, 1994.

[4] D. Baraff. Linear-time dynamics using Lagrange multipliers. In *SIGGRAPH*, pages 137–146, 1996.

[5] F. Bullo and A.D. Lewis. *Geometric Control of Mechanical Systems*. Number 49 in Texts in Applied Mathematics. Springer-Verlag, 2004.

[6] F. Bullo and A.D. Lewis. Low-order controllability and kinematic reductions for affine connection control systems. *SIAM Journal on Control and Optimization*, 44(3):885–908, 2005.

[7] T. Caldwell and T. D. Murphey. Switching mode generation and optimal estimation with application to skid-steering. *Automatica*, 2010. In Press.

[8] Sonia Chernova and Manuela Veloso. Confidence-based learning from demonstration using Gaussian Mixture Models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, 2007.

[9] M. Egerstedt, T. D. Murphey, and J. Ludwig. *Hybrid Systems: Computation and Control*, volume TBD of *Lecture Notes in Computer Science*, chapter Motion Programs for Puppet Choreography and Control, pages 190–202. Springer-Verlag, 2007. Eds. A. Bemporad, A. Bicchi, and G. C. Buttazzo.

[10] M. Egerstedt, Y. Wardi, and H. Axelsson. Optimal control of switching times in hybrid systems. In *IEEE Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, Aug. 2003.

[11] M. Egerstedt, Y. Wardi, and F. Delmotte. Optimal control of switching times in switched dynamical systems. In *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[12] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[13] K. Flaßkamp, T. D. Murphey, and S. Ober-Blöbaum. Switching time optimization in discretized hybrid dynamical systems. In *IEEE Int. Conf. on Decision and Control (CDC)*, Submitted.

[14] Daniel H. Grollman and Odest Chadwicke Jenkins. Dogged learning for robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, 2007.

[15] J. Hauser. A projection operator approach to optimization of trajectory functionals. In *IFAC World Congress*, Barcelona, Spain, 2002.

[16] E. Jochum and T. D. Murphey. A Robotic Pygmalion: Choreography for an automated marionette play. *Puppetry International*, 27:22–24, 2010.

[17] 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.

[18] E. Johnson and T. D. Murphey. Linearizations for mechanical systems in generalized coordinates. In *American Controls Conf. (ACC)*, pages 629–633, 2010.

[19] E. Johnson and T. D. Murphey. Second-order switching time optimization for nonlinear time-varying dynamic systems. *IEEE Transactions on Automatic Control*, 2010. Accepted for Publication.

[20] E. R. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, 2010.

[21] E.R. Johnson and T.D. Murphey. Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings. In *International Conference on Robotics and Automation*, Rome, Italy, 2007.

[22] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schroder, and M. Desbrun. Geometric, variational integrators for computer animation. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2006.

[23] A. Lew, J. E. Marsden, M. Ortiz, and M. West. Asynchronous variational integrators. *Arch. Rational Mech. Anal.*, 167:85–146, 2003.

[24] A. Lew, J. E. Marsden, M. Ortiz, and M. West. An overview of variational integrators. In *Finite Element Methods: 1970's and Beyond*, pages 98–115, 2004.

[25] A. Lew, J. E. Marsden, M. Ortiz, and M. West. Variational time integrators. *Int. J. Numer. Methods Engrg*, 60:153–212, 2004.

[26] J. E. Marsen and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, pages 357–514, 2001.

[27] P. Martin, E. Johnson, T. D. Murphey, and M. Egerstedt. Constructing and implementing motion programs for robotic marionettes. *IEEE Transactions on Automatic Control*, 2010. Accepted for Publication.

[28] T. D. Murphey and M. E. Egerstedt. Choreography for marionettes: Imitation, planning, and control. In *IEEE Int. Conf. on Intelligent Robots and Systems Workshop on Art and Robotics*, 2007. 6 pages.

[29] K. Nichols and T. D. Murphey. Variational integrators for constrained cables. In *IEEE Int. Conf. on Automation Science and Engineering (CASE)*, pages 802–807, 2008.

[30] R. Smith. Dynamics Simulation: A whirlwind tour (current state, and new frontiers), 2004. http://ode.org/slides/parc/dynamics.pdf.

[31] R. Smith. Open Dynamics Engine, 2008. http://www.ode.org.

[32] M. West. Variational integrators. *California Institute of Technology Thesis*, 2004.