Robotic Puppets and the Engineering of Autonomous Theater

Elizabeth Jochum*, Jarvis Schultz, Elliot Johnson, and T.D. Murphey

Northwestern University,
Evanston IL USA

Abstract. This paper outlines the design of software for embedded control of robotic marionettes using choreography. In traditional marionette puppetry, the puppets often possess dynamics that are quite different from the creatures they imitate. Puppeteers must therefore understand and leverage the inherent dynamics of the puppets to create believable and expressive characters. Because marionettes are actuated by strings, the mechanical description of the marionettes either creates a multi-scale or degenerate system—making simulation of the constrained dynamics challenging. Moreover, marionettes have 40-50 degrees of freedom with closed kinematic chains. Generating puppet choreography that is mimetic (that is, recognizably human) results in a high dimensional nonlinear optimal control problem that must be solved for each motion. In performance, these motion primitives must be combined in a way that preserves stability, resulting in an optimal timing control problem. Our software accounts for the efficient computation of the 1) discrete-time dynamics that preserve the constraints and other integrals of motion, 2) nonlinear optimal control policies (including optimal control of LTV systems), and 3) optimal timing of choreography, all within a single framework. We discuss our current results and the potential application of our findings across disciplines, including the development of entertainment robots and autonomous theater.

Keywords: Robotic Puppets, Control and Art

1 Puppets Manipulated By Machines Manipulated by Engineers

Creating autonomous machines that have artistic or aesthetic functions has been a subject of inquiry since antiquity. The merging of control theory with artistic practice has shifted the approach of artists and engineers away from creating systems that merely imitate artful or expressive gestures towards those in which the actions or behaviors are directly related to the environment and the system

* 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.

dynamics. Ideally, this research will contribute to a deeper understanding of the relationship between art and the behavior of dynamical systems. However as the experiments in this book demonstrate, it can be difficult to differentiate those features which distinguish artistic function from utilitarian function. Rather than precisely defining what makes a work of art *artful* —itself an elusive task that has occupied art historians and aesthetic philosophers for centuries [1–3] —it can be advantageous to use an established art form or practice as the basis for criteria and evaluation of experiments. The aesthetic frameworks of established art forms such as dance [4], musical composition [5], or puppetry [6] provide external referents that enable researchers to measure the degree to which systems are able to generate artistic behaviors or artifacts.

Out project uses marionette puppetry as a testbed for exploring the automated synthesis of control strategies for complex, highly-dynamic, underactuated systems. Marionette puppetry has a unique approach to creating expressive, mimetic behaviors that approximate human gestures and behaviors using a range of abstracted motions that indicate —but do not replicate —recognizably human motions. In performance, puppets acquire a grace and agility not often seen in animatronics, themselves automated systems that aim at mimicry but whose movements are typically heavy, slow, and perfunctory. The goal of our project is to emulate the control technique of human puppeteers and to develop automated puppets while maintaining the natural dynamics of marionettes. Controls strategies that preserve the marionette's dynamics are important as puppets create the illusion of life through the art of indication rather than precise mechanical reproduction. We anticipate that our robotic marionette platform will allow for a wider, more artistic range of automated motions for entertainment robots.

Artists and engineers have long experimented with developing efficient methods for simulating highly articulated rigid body systems [7–12]. However, these efforts have typically focused on designing stable physical motions rather than on *control* calculations. This paper focuses on the question of how to both simulate and control an arbitrarily complex rigid body system while maintaining scalability and convergence of the resulting numerical routines. Recognizing the potential of this research for entertainment, industrial, and medical applications, we use a robotic marionette system (seen in Fig. 1) as an example of a complex system that requires carefully embedded control that can handle many degrees of freedom. String marionettes are interesting because they partly resist the puppeteers attempts to direct them: puppeteers are forced to compromise with the dynamics of the underactuated puppet to create recognizable representations of human motion. A control-based analysis of string puppetry prompts the question of whether or not puppets can be programmed to perform autonomously—a question that has been considered with great vigor by theater artists [13–16] and is of increasing relevance given contemporary theatrical productions that combine animatronic technology with marionette control systems, such as Global Creatures' *How To Train Your Dragon* (2012) and *King Kong* (2013).

This project is a collaboration with Georgia Tech, the Atlanta Center for Puppetry Arts, and Walt Disney Imagineering/ Disney Research. Disney Imag-

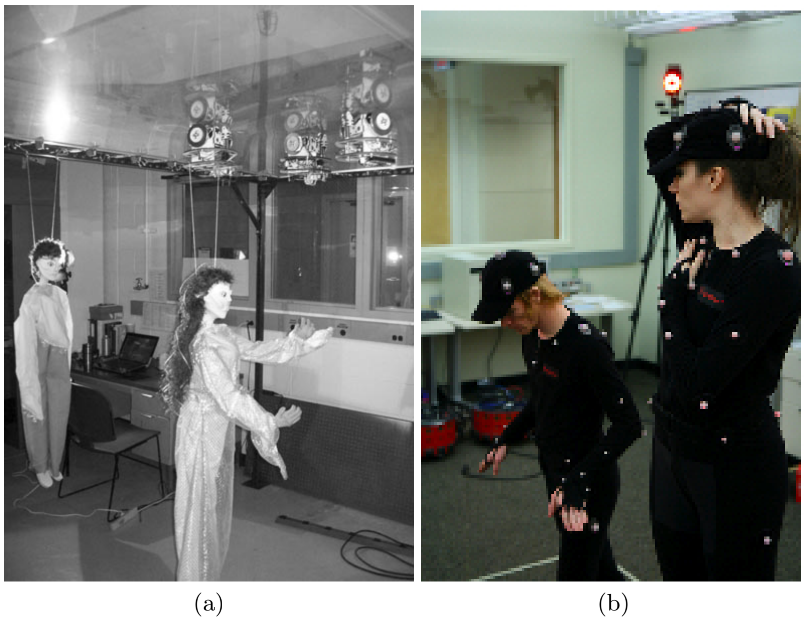


Fig. 1: The robotic marionette system in (a) is actuated by small wheeled robots that run on the underside of a tarp. The goal is to use motion captured from the dancers in (b) as reference data for the marionettes. Software must transform the dancers' motion into dynamically admissible motion for the marionettes and combine these motions together using choreography.

ineering has played a central role in developing the hardware platform. While the use of animatronics is widespread in film, museums, and theme-parks its influence on live performance has been largely negligible. The absence of animatronics from theater stages can be attributed to the tremendous technical difficulties and safety risks posed by combining massive robots alongside human performers. However the more likely explanation for their absence is that animatronics lack the human-feeling and artistry found in the live performance of direct-contact puppetry. When compared to marionettes, traditional animatronics are heavy, slow, and expensive. Robotic marionettes promise to be both more agile and less costly, and could potentially expand the possibilities for automated theatrical performances. For example Global Creatures 2013 production of the stage musical *King Kong* combined traditional puppetry techniques with automated marionette control to manipulate a six-meter tall silverback gorilla alongside human performers. While the partially-automated performance signaled a new paradigm for robotic puppets, the refined gestures and expressive behaviors still relied heavily on real-time operation by human puppeteers [17]. In this case, the reliance on human puppeteers indicates the tremendous technical challenges of automating a process that, through training, happens intuitively for humans.

Controlling marionettes is a very challenging technical problem: string marionettes have many degrees of freedom, have mechanical degeneracy due to the strings, are very lowly damped, and are highly constrained. However, human puppeteers have demonstrated a reliable ability for controlling string marionettes and solving these high-dimensional motion planning problems—puppeteers convincingly imitate human motion using marionettes—so we know the problems are solvable.

Part of this project involved the authors working with professional puppeteers to formally understand how they prepare for performance and what decisions they make during performances. Working alongside puppeteers at the Center for Puppetry Arts in Atlanta, Disney Research, and the Denver Puppet Theater provided an opportunity to discover, in practical terms, how puppeteers see their jobs as a combination of algorithmic concerns (e.g., how does one organize motion) and design concerns (e.g., how does one produce mechanical objects that are easily manipulated to produce desired motions). Puppets are complex mechanisms —controlling them involves constant trade-off between mechanical capacity and sophistication of expression. Puppeteers emphasize three phases of motion imitation —Imitate, Simplify, Exaggerate —that represent the need to capture a motion, to then simplify it for reliable mechanical reproduction, and then exaggerate it for performance. Puppeteers also make strategic decisions about how to build a puppet that will be amenable to certain motions and how to best coordinate the physical relationship between the puppeteer and the performing object. Puppeteers coordinate the timing of a motion so they can interact with other puppeteers, sometimes collaborating to control a single marionette or groups of puppets, ensuring that the marionettes remain animated throughout the performance. Scripts of puppet plays describe the action using four parameters: temporal duration, agent, space, and motion (i.e., when, who, where, and what). These motions are grouped and executed according to counts that specify when each motion begins and ends. During rehearsals and performance, the puppeteer makes decisions about the use of force, dynamics, and movement qualities that determine the expressive characteristics and the overall visual effect, handling complex choreographic sequences and solving problems of uncertainty, often before they arise. These are the processes that we set out to understand.

Using a control-based analysis, we aim to understand how puppeteers manage complexity and uncertainty and apply these insights to autonomous theater productions and optimization problems in other critical areas. Our current hypothesis is that choreography plays a critical role in how puppeteers manage complexity, and that the study of marionette choreography will further our understanding of other complex applications, such as embedded control of prosthetics (briefly discussed at the end of this article). Our goals are to synthesize motion control for complex systems. To that end, choreography provides a way of categorically identifying useful and recognizable motions (e.g., walking, running, waving, reaching) that can be combined together in phrases. How one combines and orders these motions determines the stability and smoothness of transitions between motions.

This paper is organized as follows: Section 2 introduces marionette puppetry and articulates the distinct technical challenges of generating recognizable and artistic motions using marionettes. Section 3 describes typical analytical approaches in dynamic simulation and optimal control and the specific software requirements these approaches create. Section 4 discusses which special considerations should be taken into account when working in discrete time. Section 5 illustrates our software approach and briefly discusses examples of systems that the software automatically optimizes successfully. Section 6 discusses the relevant features of the current framework and the broader impacts for automated puppetry and the engineering of autonomous theater.

2 Puppets: Aesthetic and Mechanical Considerations

From antiquity to the present, artists and engineers have sought to create mechanical figures that generate expressive and lifelike behaviors [18, 13]. Puppets are part of this lineage, and are representatives *par excellence* of humanity's insatiable thirst for bringing ordinary object to life through motion. As art and technology scholar Chris Salter has observed, the histories of performing machines and robotic art have often involved a continual mingling between mimetic aspects (that is, objects that are imitative of lifelike behavior in appearance) and machinic aspects (electromechanical behavior that, though animate, is not anthropomorphic) rather than a disambiguation [19]. That is, human artists are interested in designing machines that emulate aspects of human creativity. Before the advent of computing and electronics, the approach was to design objects that appeared lifelike and imitated human actions, such as the 18th century humanoid automata built by Jacques de Vaucanson and Henri-Louis and Pierre Jaquet-Droz [20]. Puppets are the progenitors to these and other attempts to create mechanical life, and are part of a lineage that extends to present-day animatronics.

Puppets can take the shape of realistic or abstract figures, and are designed for use in theatrical settings. A puppet is generally defined as a material object that makes temporal use of sources of power that exist outside of itself, and that are not its own attributes [13, 21]. Animatronics are puppets that are electronically controlled through various actuators (electrical, hydraulic, pneumatic), however the absence of a human performer and repetitive or open-loop performance means that they are not typically regarded in the same way that human-powered puppets are. Setting aside for a moment the question of human agency, we recognize that the main purpose of a puppet is movement: to establish a meaningful presence the puppet relies on motions to create a character or presence that is both recognizable to the spectator and conveys a certain artistic truth. The puppet's power of expression is therefore not determined by how well it precisely mimics human behavior, but rather by its ability to abstract human motions and offer an artistic projection of those motions and behaviors. In other words, the goal of puppetry motion is not to copy but to *create*. As Kingston et al. have also observed, a puppets primary purpose is to communicate through motion [6].

Puppeteers are extremely inventive, and have developed numerous methods for controlling (often called "manipulating") and constructing expressive, moving objects. String marionettes are dynamically unique among puppets, and more than other types of puppets have the potential to teach us about optimal control. Unlike glove puppets or rod puppets which are controlled through direct, corporeal contact (hands-on, hands-in), string marionettes are operated from above by a varying number of strings. For humanoid puppets, these strings are usually attached to the puppet's head, torso, shoulders, arms, and legs, and can be strung to generate specific actions based on the specifications of the choreography. The strings can be manipulated with a variety of controls—the most common is the multi-stringed wooden control or "airplane" (crossbar) mechanism. This technique is used in Asian and European puppetry forms, and puppeteers can opt to vary the string lengths and attachment points according to the needs of each production or scene. The precision of the puppet's motions is relative to the control of the figures—the longer the strings, the weaker the impetus, which results in softer, less precise movements [13]. Puppeteers can control marionettes at ground level working alongside the puppet and in full view of the audience, or from a position high above the stage from a bridge and out of the audience's sight lines. In either arrangement, the puppeteer must learn to balance the dynamics of the puppet against the need to execute expressive choreography that convincingly imitates—but does not replicate—human motions. Because string marionettes resist the puppeteer's attempts to direct them, puppeteers have developed highly-evolved strategies for generating sequences of complex motions—this is what is known as marionette choreography. Because of their dynamics, string marionettes are arguably the most difficult type of puppet to operate. This also makes them a good testbed for a control-based analysis.

3 Typical Approach

In this section we discuss typical analytical approaches to optimal control and the types of software infrastructure these approaches assume. We start with a discussion of how one might describe the dynamics of a mechanical system and then discuss computing optimal controllers for that system.

3.1 Dynamics

Marionettes are subject to the physics of the world: they swing and sway according to the forces of gravity and the interplay between the different bodies —individual units such as the torso, forearms, and legs —in the marionette. This interplay between the bodies falls within the realm of dynamics and simulation, where the dynamic description comes from a physics-based understanding of puppet motion. When computing dynamics we are typically trying to compute equations of 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 a typical humanoid marionette, for instance, the marionette alone (no actuators) has 10 rigid bodies, so the state space would be more than 120 dimensions. If one includes string actuation in the degree count, that adds one degree of freedom for each string's length and an additional element of $SE(3)$ for each string endpoint (treating the inputs as "kinematic inputs" [22]). For a typical marionette with 6 strings of variable lengths, this brings the total nominal dimension of the state space up to $12 \cdot 10 + 2 \cdot 6 + 12 \cdot 6 = 204$. Naturally, we don't want to be solving for feedback controllers in a 204 dimensional space if it can be avoided. To avoid such high dimensions, we

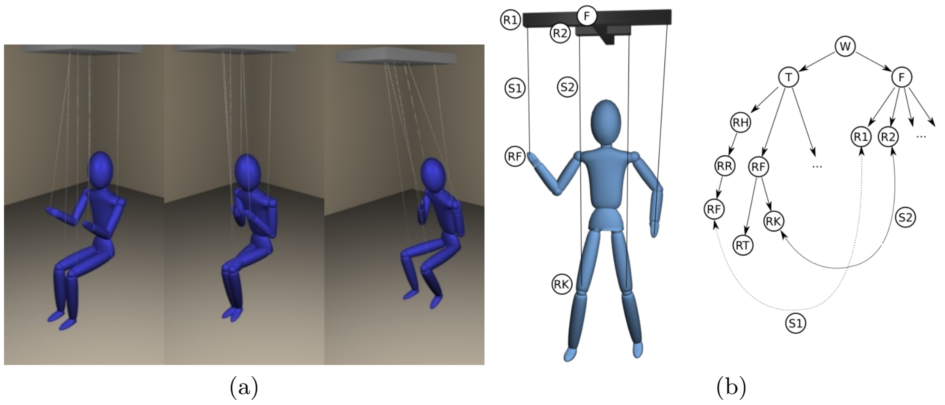


Fig. 2: Simulation of complex rigid bodies can take advantage of the mechanical topology of the system. For instance, a marionette is being simulated in (a) using a tree structure representation of the humanoid form in (b) and representing the constraints by cycles in the graph (see [23]).

don't want to represent Eq. (1) as Newton-Euler equations and instead insist on working in generalized coordinates. In the case of the marionette, this reduces the dimension of the state to $2m$, where m is the number of generalized coordinates. This yields 22 configuration variables for the marionette itself and another 18 degrees of freedom for the string actuators (6 strings with endpoints moving in $\mathbb{R}^2 \times \mathbb{R}$), yielding 80 states. By utilizing a kinematic reduction [24, 25] we can reduce the state of the actuators down to 18 because they are fully actuated.

This gives us equations of motion of the following form:

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

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 [22].) This leaves us with a much smaller, more manageable system to work with that only has a total of 62 dimensions in its state space. Crucially, the strings are modeled as holonomic constraints, relating the lengths of the strings to the dynamic configurations, of the form

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

which must be maintained during simulation and control. Assuming for the moment that Eqs (2) and (3) can be stably simulated in an efficient manner, how do we then construct the differential equation and constraints in a systematic manner? The standard way to do this is based on Featherstone's early work [12] on articulated body dynamics. This work was largely used in the context of animation and digital puppetry, where the requirements are substantially different than embedded control. For example, animation requires a simulation to "look right" only once, while controlled physical systems must be repeatable. Recursive approaches to calculating dynamics [26, 12] take advantage of special representations of mechanical systems that allow the values needed for simulation to be calculated quickly and avoid redundant calculations. The work in this paper is based on the methods presented in [23] (based on [12]). Here, 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 rotations 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 2 is an example of a simulated 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 calculations of $f(\cdot, \cdot)$ in Eq. (2) and $h(\cdot)$ in Eq. (3). Moreover, one can use the same structure to efficiently calculate the linearization [27], which is critical to nonlinear optimal control calculations, discussed in the next section.

3.2 Nonlinear Optimal Control

Controlling marionettes involves choosing how the strings must be operated in order to achieve some desired trajectory. For instance, if we wish a marionette to generate a walking motion, the strings must pull the limbs in such a way that the body of the marionette indicates walking. The choice of string motions will, of course, depend on the physics-based description of the marionettes. For instance, if the masses of the legs are very high, we might have to pull on the strings differently than if the masses were very low. Optimal control plays a significant role in determining the outcome because it provides an algorithmic means of choosing string lengths and endpoint positions in a manner that takes the physics-based description of the marionettes into account. Optimal control typically starts out with a cost function of some sort, often of the form

$$J = \int_{t_0}^{t_f} L(x(t), x_{\mathrm{ref}}(t), u(t))dt + m(x(t_f), x_{\mathrm{ref}}(t_f)) \tag{4}$$

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) such as in the imitation problem mentioned above. We can minimize this cost function subject to the dynamics in Eqs. (2) and (3) by 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 [28]. The projection operator $\mathscr{P}(\cdot)$ comes from computing a feedback law (discussed in more detail in the next section). 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 must only solve a standard time-varying LQR problem. This means that one must be able to compute the time-varying linearization

$$\dot{z} = A(t)z + B(t)v \tag{5}$$

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 the descent direction involves computing the Riccati equations

$$\dot{P} + A(t)^T P + PA(t) + Q - PB(t)R^{-1}B(t)^T P = 0. \tag{6}$$

If we additionally want to guarantee quadratic convergence, then we can find a descent direction by solving a different LQR problem

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

where

$$\begin{aligned} D^2J(\xi)\cdot(\zeta^1,\zeta^2) = D^2g(\xi)\cdot(D\mathscr{P}(\xi)\cdot\zeta^1, D\mathscr{P}(\xi)\cdot\zeta^2) \\ +Dg(\xi)\cdot D^2\mathscr{P}(\xi)\cdot(\zeta^1,\zeta^2). \end{aligned} \tag{7}$$

The second derivative $D^2\mathscr{P}(\cdot)$ requires that we be able to also calculate $\frac{\partial^2 f}{dx^2}$, $\frac{\partial^2 f}{du^2}$, and $\frac{\partial^2 f}{dxdu}$. The details of this approach can be found in [28] and elsewhere, but for our purposes we should be able to compute Eqs. (2), (3), (4), (5), (6), and (7) in software. The difficulty of this approach is that we are representing the optimal control problem in continuous time while the actual computations are in discrete time. We will discuss this further in Section 4 and we see that it is only when we perform optimal control calculations in discrete time that we arrive at convergence of the algorithms that provide motion imitation.

3.3 Choreography and Hybrid Optimal Control

In [29–31] we developed an optimal control interpretation of puppet choreography. In particular, we formalized choreography as a sequence of *modes* that can be pieced together to form a script of motions. 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. For a gradient descent algorithm, one must be able to compute the derivative $\frac{\partial J}{\partial \tau_i}$ —the derivative of the cost function with respect to the switching times—which depends on the switching time adjoint equation

$$\dot{\rho} + A(t)^T\rho + \frac{\partial L}{\partial x} = 0 \tag{8}$$

along with a boundary condition at $\rho(t_f)$ (see [32–35]). This adjoint equation only needs to be computed once to compute all the derivatives of J. If one wants to compute the second derivative of J, e.g. to utilize Newton's method, then the second-order switching time adjoint equation

$$\dot{P} + A(t)^TP + PA(t) + \frac{\partial^2 L}{\partial x^2} + \sum_k \rho_k \frac{\partial^2 f^k}{\partial x^2} = 0 \tag{9}$$

along with its boundary condition $P(t_f)$ must be solved [33]. This adjoint equation, along with a solution to Eq. (8), needs to be computed only once to obtain all the derivatives of J. Note that the second-order switching time adjoint equation is the same as the Riccati equation in Eq. (6) except that the Riccati equation has a different final term. Indeed, both Eq. (8) and Eq. (9) only require first and second derivatives of f_i with respect to the state, so those are all that are needed for software; hence, the choreographic optimization requires the same software capabilities as the smooth optimization described in Section 3.2.

4 Discrete time with scalability

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. (2) and (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, this often creates a multi-scale simulation problem that is incompatible with real-time operation. As an alternative, we consider variational integrators [36–42]. 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 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

$$D_1 L_d(q_k, q_{k+1}, k) + D_2 L_d(q_{k-1}, q_k, k) = F_k \tag{10}$$

$$h(q_{k+1}) = 0 \tag{11}$$

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}. Alternatively, one can utilize the discrete Legendre transform to define the discrete generalized momentum p_k, and then convert the root solving problem of Eq. 11 into a one-step rootsolving problem where given the pair (q_k, p_k) one implicitly solves for the next timestep pair (q_{k+1}, p_{k+1}). Repeating this rootsolving procedure forms the basis of simulation. Using this method, we can (using a recursive tree description similar to the one described in Section 3.1), simulate the marionette in 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 [43]. Once we have made a choice of state (we choose $x_k = (q_k, p_k)$), 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 in principle no different in the variational integrator case than it is in the standard ODE case.

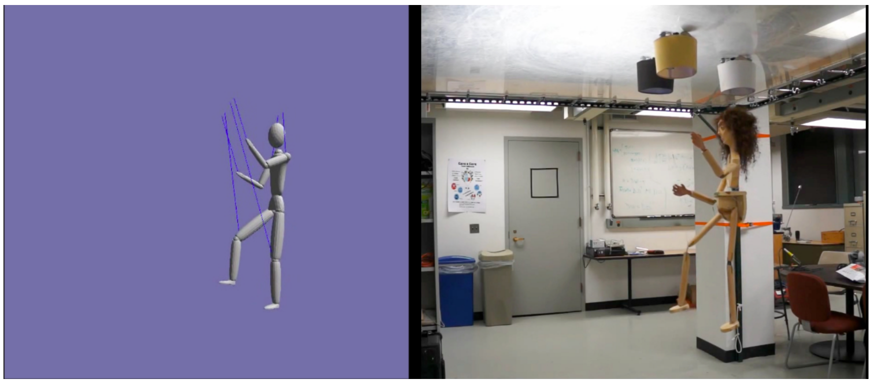


Fig. 3: Software using variational integrators as the basis for simulation can accurately predict marionette motion using comparatively large time steps. In this case, the time step is $dt = 0.01s$.

More importantly, the fact that f_k is implicitly defined by the DEL equations does not affect whether the linearization is implicitly defined. In fact, one can calculate an *exact* linearization of the DEL equations, including constraints and closed kinematic chains [27]. So we may wish to know what the discrete-time version of Eqs. (6) and (7) are. (These can be found in [44].) The difficulty is that one cannot linearize Eq. (1) directly because that is the infinitesimal linearization; to get a discrete-time linearization one would nominally have to solve the state transition matrix (STM) locally over the time step. Approximating the STM leads to a linearization that does not respect the constraints, leading to a local optimal controller that essentially fights the constraints. In contrast, taking variations directly with respect to x_k yields an algebraic calculation for the linearization and higher-order derivatives with respect to the state. As with variational integration, the key to linearization is to take variations with respect to the discrete state rather than the continuous state. For nonlinear optimal control in the discrete time setting, we need to know if the resulting projection operator (as discussed in Sec. 3.2) is in fact a projection and whether it is differentiable. To see that such a projection is valuable, consider Fig. 4 [45], where a planar double pendulum trajectory is being optimized. The initial guess for the optimal solution is the "zero" solution, the optimal solution is the solid black line, and the *first* iteration of Newton's method using the projection operation is the dotted line. Hence, one step of Newton's method almost solves the global optimization in this case. Naturally, that will not always be the case, but this is an indication of how much generating a differential projection operation can help. Let $\overline{\xi} = (\overline{x}, \overline{u})$ be a desired, potentially infeasible, curve in the space the trajectories reside in and let $\xi = (x, u)$ be feasible trajectories. The continuous

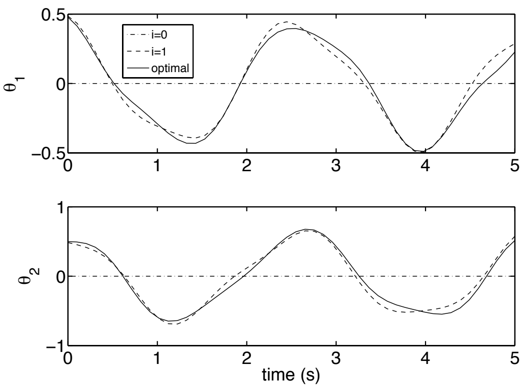


Fig. 4: Projection-based variational optimization of a planar double pendulum [45]

time projection operator is defined by $\xi = \mathscr{P}(\bar{\xi})$ such that

$$\begin{aligned} x(t_0) &= \overline{x}(t_0) \\ \dot{x} &= f(x,u) \\ u &= \overline{u} - K(t)(x - \overline{x}) \end{aligned}$$

where the feedback gain $K(t)$ comes from solving the Riccati equation in Eq. (6). One can verify that $\mathscr{P}(\cdot)$ is a projection and that it is C^∞ if f is C^∞. What do we do if we are using Eqs. (10) and (11) instead of Eqs. (2) and (3)? Then the discrete projection operator $\mathscr{P}_d(\cdot)$ is $\xi_d = \mathscr{P}_d(\overline{\xi}_d)$ such that

$$\begin{aligned} x_0 &= \overline{x}_0 \\ x_{k+1} &= f_k(x_k, u_k) \\ u_k &= \overline{u}_k - K_k(x_k - \overline{x}_k) \end{aligned} \tag{12}$$

where the discrete time feedback gain K_k comes from solving a discrete time Riccati equation. To see that it is a differentiable projection, we introduce the following Lemma.

Lemma 1. *$\mathscr{P}_d(\cdot)$ is a projection.*

Proof. We need to show that the projection satisfies the property $\mathscr{P}_d(\bar{\xi}_d) = \mathscr{P}_d(\mathscr{P}_d(\bar{\xi}_d))$. First we calculate several terms of $(a,b) = \mathscr{P}_d(\alpha,\mu)$ and get $a_0 = \alpha_0$, $b_0 = \mu_0 - K_0(a_0 - \alpha_0) = \mu_0$, $a_1 = f_0(a_0,b_0) = f_0(\alpha_0,\mu_0)$, and $b_1 = \mu_1 - K_1(a_1 - \alpha_1)$. Now calculate several terms of $(x,u) = \mathscr{P}_d(a,b)$ and find that $x_0 = a_0 = \alpha_0$, $u_0 = b_0 - K_0(x_0 - a_0) = b_0 = \mu_0$, $x_1 = f_0(x_0,u_0) = f_0(\alpha_0,\mu_0) = a_1$, and $u_1 = b_1 - K_1(x_1 - a_1) = b_1 - K_1(a_1 - a_1) = b_1$. By induction we find $\mathscr{P}_d \circ (\alpha,\mu) = \mathscr{P}_d \circ \mathscr{P}_d \circ (\alpha,\mu)$. Hence, $\mathscr{P}_d(\cdot)$ is a projection operation from discrete-time representations of curves $\overline{\xi}_d$ to discrete-time representations of trajectories ξ_d.

Next we need to calculate the derivative of $\mathscr{P}_d(\cdot)$, starting with $\xi_d = \mathscr{P}_d(\bar{\xi}_d)$ (we are going to drop the d from ξ_d for notational convenience).

$$\delta\xi = D\mathscr{P}_d(\bar{\xi}) \circ \delta\bar{\xi}$$

So, by Eq. (12), we get

$$\begin{aligned}
\delta x_0 &= \delta\bar{x}_0 \\
\delta x_{k+1} &= \frac{\partial f_k}{\partial x_k}\delta x_k + \frac{\partial f_k}{\partial u_k}\delta u_k = Df_k \circ \delta\xi_k \\
\delta u_k &= \delta\bar{u}_k - K_k(\delta x_k - \delta\bar{x}_k).
\end{aligned}$$

where $\frac{\partial f_k}{\partial x_k}$ is shorthand for $\frac{\partial f_k}{\partial x}(x_k, u_k, k)$. (The same applies to $\frac{\partial f_k}{\partial u_k}$ and Df_k.) As in the continuous case, the derivative of the discrete projection is a discrete linear system. The second derivative is also straightforward (here we let $\delta\bar{\xi}^1$ and $\delta\bar{\xi}^2$ be two independent perturbations to $\bar{\xi}$).

$$\delta^2\xi = D^2\mathscr{P}(\bar{\xi}) \circ (\delta\bar{\xi}^1, \delta\bar{\xi}^2)$$

which implies, again by Eq. (12), that

$$\begin{aligned}
\delta^2 x_0 &= 0 \\
\delta^2 x_{k+1} &= D^2 f_k \circ (\delta\xi_k^1, \delta\xi_k^2) + Df_k \circ \delta^2\xi_k \\
&= \frac{\partial f_k}{\partial x_k}\delta^2 x_k + \frac{\partial f_k}{\partial u_k}\delta^2 u_k + D^2 f_k \circ (\delta\xi_k^1, \delta\xi_k^2) \\
\delta^2 u_k &= -K_k\delta^2 x_k.
\end{aligned}$$

Rewriting the second derivative, we get:

$$\begin{aligned}
\delta^2 x_{k+1} &= \frac{\partial f_k}{\partial x_k}\delta^2 x_k + \frac{\partial f_k}{\partial u_k}\delta^2 u_k + D^2 f_k \circ (\delta\xi_k^1, \delta\xi_k^2) \\
&= \left[\frac{\partial f_k}{\partial x_k} - \frac{\partial f_k}{\partial u_k}K_k\right]\delta^2 x_k + D^2 f_k \circ (\delta\xi_k^1, \delta\xi_k^2).
\end{aligned}$$

This is a discrete affine system, equivalent to a discrete linear system with an input:

$$\begin{aligned}
\delta^2 x_{k+1} &= \hat{A}_k\delta^2 x_k + \hat{B}_k \\
\hat{A}_k &= \left[\frac{\partial f_k}{\partial x_k} - \frac{\partial f_k}{\partial u_k}K_k\right] \qquad \hat{B}_k = D^2 f_k \circ (\delta\xi_k^1, \delta\xi_k^2).
\end{aligned}$$

Hence, the projection operation $\mathscr{P}_d$ is twice differentiable with derivatives that are represented by discrete-time linear difference equations, allowing us to apply Newton's method to optimal control problems.

5 Examples

We now consider the model of a humanoid marionette shown in Fig. 5. The marionette has 40 configuration variables and is actuated by 6 strings. The strings

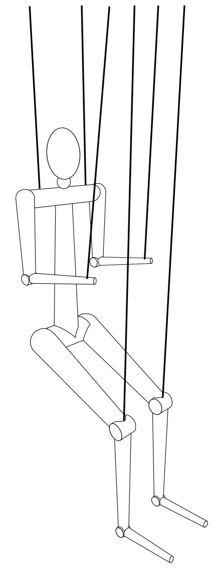
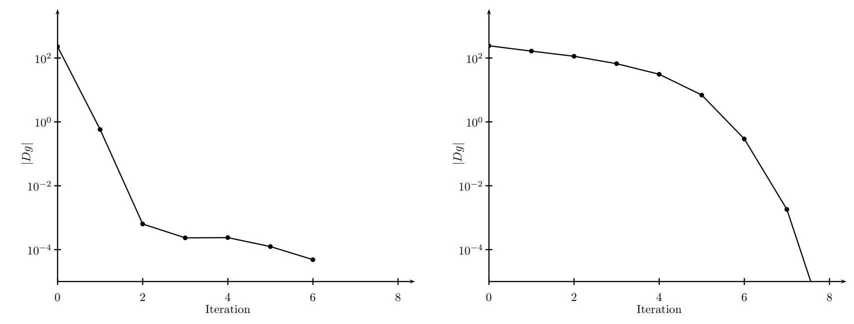


Fig. 5: The 3D Marionette is actuated by six strings.

are modeled as holonomic constraints. Kinematic configuration variables—those that we assume satisfy first-order dynamics where we control the velocity of the variable directly—control the two-dimensional position of the end-point of each string as well as the string length. There are no joint torques and only slight damping applied to each dynamic configuration variable equally. In order to indicate the real-time feasibility of the algorithms discussed in the previous sections, we now provide some timing data for the marionette simulation in Fig. 5 we have a system that has 22 dynamic degrees-of-freedom, 18 kinematic degrees of freedom (corresponding to the actuation of the strings), and 6 total holonomic constraints. To evaluate the continuous dynamics that would be used in a standard integrator, one evaluation of $f(x, u)$ requires 2.7 ms, while the first derivative with respect to the state (i.e., the linearization) requires 24 ms and the second derivative with respect to the state requires 400 ms. Note that this does not indicate how long it will take to simulate a particular length of time because the time step is not included here as we are not working in discrete time. With the variational integrator from Eqs. (10) and (11) with a time step of 0.01 (no other parameters are needed when using variational integrators, even with the degeneracies and constraints the strings introduce), the update step requires 5.53 ms while the first derivative with respect to the state (i.e., the exact discrete linearization) takes 2.4 ms and the second derivative with respect to the state takes 130 ms. This means that, at minimum, we can simulate and evaluate controllability in real-time. Two optimizations are discussed in the following subsections. The optimization in Sec. 5.1 uses a desired trajectory that was generated separately by simulating the system. Sec. 5.2 discusses an optimization with a reference generated from motion capture data.

5.1 Desired Motion: Simulated Trajectory

A desired trajectory was generated by simulating the system forward in time. The lengths of the arm and leg strings were varied sinusoidally to create a “walking” motion. The configurations of the trajectory were saved. The rest of the state (e.g, configuration velocity or discrete momentum) and the simulation inputs were discarded and replaced with uniformly zero trajectories. This results in a smooth desired trajectory that we expect the puppet to be able to track, but remains an infeasible trajectory. The marionette was optimized to the desired trajectory in both continuous and discrete time. Both optimizations successfully converged to solutions that track the desired configuration very well. Convergence plots for both optimizations are shown in Fig. 6. The source code for the discrete optimization is distributed with `trep` at *http://trep.googlecode.com* in the file `/examples/puppet-optimization.py`. Fig. 6 shows that the continu-



(a) Continuous Time (b) Discrete Time

Fig. 6: Convergence plots for continuous and discrete time optimizations from a simulated desired trajectory.

ous optimization initially converges drastically faster than the discrete one: it tracks the desired trajectory almost perfectly after a single step. The discrete optimization makes slow progress initially but converges quickly after about five iterations. The discrete optimization takes 5:50s to finish. Each iteration takes between 15 and 60 seconds depending on the descent direction type and number of Armijo steps. Although the convergence plot is flattering for the continuous optimization, there were numerous problems. Unlike the discrete optimization, the continuous optimization was highly sensitive to the optimization parameters. Large ratios between the maximum and minimum state cost cause the optimization to fail. Additionally, for a successful continuous optimization, the terminal conditions must be significantly relaxed. The discrete time optimization suffers from neither of these problems.

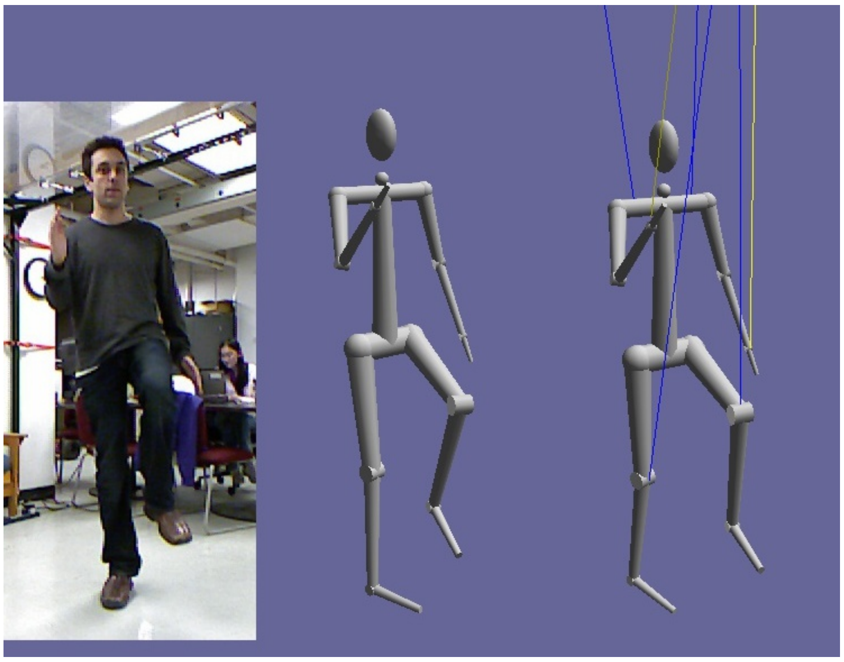


Fig. 7: These three images show a single frame from the motion-capture optimization. The left-most picture shows an RGB image recorded by a Microsoft Kinect®. The middle figure is the motion capture data found from the Kinect's depth map. The right-most figure is the optimized trajectory.

5.2 Desired Motion: Motion Capture Data

A more practical application of the trajectory optimization is finding trajectories to track data acquired from a motion capture system. In this example, a desired trajectory was generated using a Microsoft Kinect® to record a student walking in place. This process is illustrated in Fig. 7. In this case, the continuous optimization was unable to converge. The discrete optimization converged successfully and found a trajectory that closely approximates the student's movement. Figure 8 plots the desired trajectory and optimization result for the angle of the right elbow as an example. The trajectory found by the optimization tracks the desired trajectory very well. However, a large amount of noise was introduced. This is most likely caused by too large of a ratio between the weight of the configuration portion of the state compared the discrete momentum portions and the cost of the inputs.

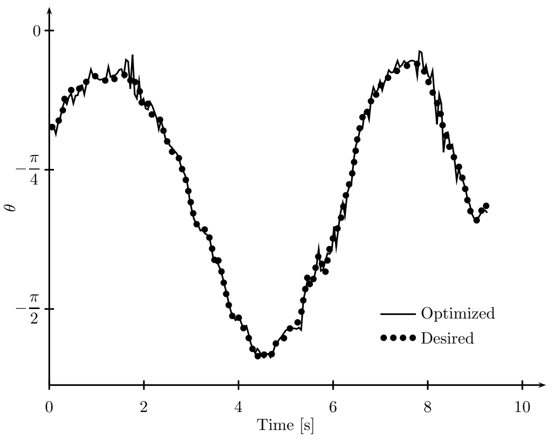


Fig. 8: The optimal trajectory for the right arm elbow tracks the desired trajectory well.

6 Conclusions

The robotic marionette project is an example system that forces us to create software that can both simulate and control complex mechanical systems. The marionettes play a vital role in driving the system development—they have mechanical degeneracies, closed kinematic chains, and are high dimensional, but despite these features puppeteers are able to successfully and reliably control them. Therefore, marionettes make a good testbed for understanding whether or not our software is producing reasonable results. Evaluating the efficacy of the control system can be determined immediately by observing whether or not the motions and choreographic phrases are recognizable, smooth, and approximate the reference data. Viewed this way, we see how a control-based analysis of an existing art form allows us to conceptualize new approaches in optimal control, and also increases the likelihood that such a system will be used by artists to develop choreography for marionettes or other artificial, articulated bodies. The techniques we use provide both optimal trajectories and control laws that help stabilize those trajectories. Moreover, because we formulate the optimal control problem using a differentiable projection, we can analytically guarantee quadratic convergence locally around the optimal trajectory. It is also important to note that the techniques we have applied to the marionettes are also applicable to many other fields. For example, we have used these software techniques for the tendon-articulated hand in Fig. 9 and can compute linearizations and local LQR controllers for the hand. (This simulation capability is now being used with prosthetic control in a collaboration with the Rehabilitation Institute of Chicago.)

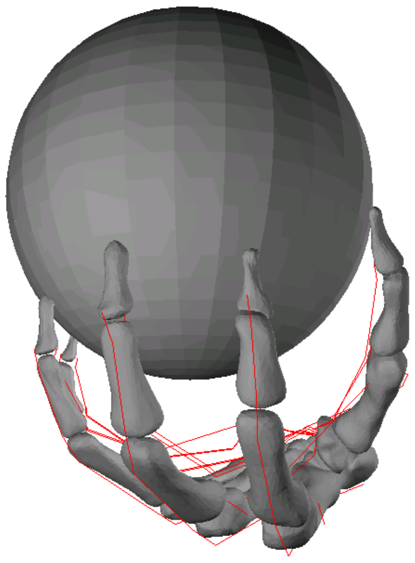


Fig. 9: The graph-based approach to calculating linearizations scales to complex mechanical systems like this dynamic model of a tendon-articulated human hand holding an object. The linearization at this configuration shows that the system is locally controllable.

Puppeteers use marionettes in dynamic, expressive ways, so we presume that extremely conservative motions based on a “quasi-static” approach or an inverse-kinematics approach are unlikely to produce interesting, artistic motions. Puppetry privileges imitation over precise replication, but to be considered “artistic” the imitation must rise above perfunctory or routine motions. Typically, the artistic quality of puppetry has been assumed to be the result of the interpretation and execution of the human puppeteer, but our automated marionette platform provokes the question of whether or not human operation is essential to a marionette’s expressiveness and theatrical value. Although we can produce an optimal “imitation” of a human motion in the case of a simple walking motion captured by the Kinect sensor, we must first solve the calculations for the full marionette for a variety of motions before we can claim that marionette imitation can be defined as an optimization problem. We are currently working on solving these problems. It remains to be seen whether or not a fully automated marionette can execute choreography in such a way that transcends mere imitation and achieves artful or aesthetic resonance.

One potential application of the automated marionettes is live entertainment and theater productions that utilize large-scale animatronics and automated performers. While Disney pioneered the technologies of animatronics, enabling artists and engineers to partially realize their shared vision of creating an autonomous theater, Creature Technology Company and Global Creatures are developing sophisticated animatronics for use in live performance. The 2012 production based on Dreamworks' animated film *How To Train Your Dragon* and the 2013 *King Kong* are two recent productions that combine large scale animatronics with human performers in a live production. Unlike traditional animatronics, these shows are purposefully designed for international tours. The demands of an international touring production present considerable challenges: the animatronics must be able to perform reliably in a wide range of venues, the choreography of these productions brings live actors into close contact with heavy, dangerous machines, and audiences expect the machines to be as interactive and agile as live performers. Perhaps not surprisingly, Global Creatures has opted to work with expert puppeteers to develop choreography for large scale animatronics in the shape of flying dinosaurs, dragons, and a six-meter tall silverback gorilla. The machines are controlled through a combination of marionette-automation and tele-operated controls (known at Global Creatures as "voodoo puppeteering") [46]. A team of puppeteers works together to control a single puppet (sometimes with a puppeteer seated in a chassis inside of the puppet), while the marionette-automation enables these large puppets to execute expressive choreography such as flying and aerial stunts—even scaling the side of a replica of the Empire State building. This unique hybrid control system expands the range and quality of motions that are available to the puppet, allowing the puppeteer to develop choreography on a larger scale than has previously been imaginable: animatronic choreography can now utilize the entire stage space. The level of automation and sensing technologies used to control the puppets further distance the human operator from the physical act of puppeteering, resulting in increasingly automated performances that are not rote or perfunctory but are rather perceived as lively and entertaining. This arrangement challenges the notion that autonomous theater can only be pre-programmed or repetitive, and reignites the debate of whether or not human operators are essential for live theater performances.

In his famous essay *What is Art?* the Russian novelist Leo Tolstoy wrote "To evoke in oneself a feeling one has once experienced and having evoked it in oneself then by means of movements, lines, colours, sounds, or forms expressed in words, so to transmit that feeling that others may experience the same feeling—this is the activity of art." As artists and engineers focus their attention on emulating artistic process through the generation of performances and artifacts based on system dynamics and control theory, established art forms such as music, dance, and puppetry provide important frameworks for modeling and evaluating the aesthetic or artistic outcomes. The research projects discussed in this volume are illustrative of the urge to understand and emulate the intangible aspects of aesthetics while evaluating the tangible outcomes of these investigations. Whether it is through the creation of aesthetic or communicative gestures [6], generating appealing or interesting musical compositions [5], or expressive and engaging choreography [4], the goal is to generate aesthetic behaviors that are emergent and are evocative of some human feeling. Whether the application of control theory can engender similar aesthetic responses as works of art generated by human artists remains to be seen, but it is a question worth considering.

References

1. N. Goodman, *Ways of Worldmaking.* Indianapolis, IN: Hackett Publishing, Jan. 1978.
2. I. Kant, *Critique of Aesthetic Judgement (1790).* Oxford, UK: Oxford University Press, Aug. 1911.
3. F. Schiller, *On the Aesthetic Education of Man.* Oxford, UK: Oxford University Press, 1794.
4. A. P. Schoellig, H. Siegel, F. Augugliaro, and R. D'Andrea, "So you think you can dance? rhythmic flight performances with quadrocopters," in *Control and Arts*, A. LaViers and M. Egerstedt, Eds. Springer-Verlag, 2013.
5. C. Huepe, M. Colasso, and R. F. Cádiz, "Generating music from flocking dynamics," in *Control and Arts*, A. LaViers and M. Egerstedt, Eds. Springer-Verlag, 2013.
6. P. Kingston, J. von Hinezmeyer, and M. Egerstedt, "Metric preference learning with applications to motion imitation," in *Control and Arts*, A. LaViers and M. Egerstedt, Eds. Springer-Verlag, 2013.
7. R. Smith, "Open Dynamics Engine," 2008, http://www.ode.org.
8. ——, "Dynamics Simulation: A whirlwind tour (current state, and new frontiers)," 2004, http://ode.org/slides/parc/dynamics.pdf.
9. D. Baraff, "Linear-time dynamics using Lagrange multipliers," in *SIGGRAPH*, 1996, pp. 137–146.
10. ——, "Fast contact force computation for nonpenetrating rigid bodies," in *SIGGRAPH*, 1994.
11. ——, "Non-penetrating rigid body simulation," in *State of the Art Reports*, 1993.
12. R. Featherstone, *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.
13. P. Francis, *Puppetry.* Palgrave Macmillan, 2012.
14. H. Kleist, "On the marionette theatre," *The Drama Review*, vol. 16, no. 3, pp. 22–26, 1972.
15. E. G. Craig, *On the Art of the Theatre.* Routledge, 2009.
16. S. Kaplin, "A puppet tree: A model for the field of puppet theatre," *TDR*, vol. 43, no. 3, pp. 28–35, 1999.
17. E. Jochum, "King kong," *Theatre Journal*, Dec. 2013, Johns Hopkins University Press.
18. J. Burnham, *Beyond modern sculpture: the effects of science and technology on the sculpture of this century.* New York, NY: George Braziller, June 1968.
19. C. Salter, *Entangled: Technology and the Transformation of Performance.* MIT Press, 2010.
20. G. Wood, *Edison's Eve: a magical history of the quest for mechanical life.* New York, N.Y.: A.A. Knopf, 2002.
21. H. Jurkowski, *Aspects of Puppet Theatre.* Puppet Centre Trust, 1988.
22. E. Johnson and T. Murphey, "Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings," in *International Conference on Robotics and Automation*, Rome, Italy, 2007.
23. E. R. Johnson and T. D. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *IEEE Transactions on Robotics*, 2010.

24. F. Bullo and A. Lewis, "Low-order controllability and kinematic reductions for affine connection control systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 3, pp. 885–908, 2005.
25. ——, *Geometric Control of Mechanical Systems*, ser. Number 49 in Texts in Applied Mathematics. Springer-Verlag, 2004.
26. Y. Nakamura and K. Yamane, "Dynamics computation of structure-varying kinematic chains and its application to human figures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 2, 2000.
27. E. Johnson and T. D. Murphey, "Linearizations for mechanical systems in generalized coordinates," in *American Controls Conf. (ACC)*, 2010, pp. 629–633.
28. J. Hauser, "A projection operator approach to optimization of trajectory functionals," in *IFAC World Congress*, Barcelona, Spain, 2002.
29. 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.
30. M. Egerstedt, T. D. Murphey, and J. Ludwig, *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2007, vol. TBD, ch. Motion Programs for Puppet Choreography and Control, pp. 190–202, eds. A. Bemporad, A. Bicchi, and G. C. Buttazzo.
31. 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.
32. 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.
33. T. Caldwell and T. D. Murphey, "Switching mode generation and optimal estimation with application to skid-steering," *Automatica*, 2010, in Press.
34. 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.
35. 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.
36. E. Johnson and T. D. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1249–1261, 2009.
37. K. Nichols and T. D. Murphey, "Variational integrators for constrained cables," in *IEEE Int. Conf. on Automation Science and Engineering (CASE)*, 2008, pp. 802–807.
38. 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.
39. A. Lew, J. E. Marsden, M. Ortiz, and M. West, "Variational time integrators," *Int. J. Numer. Methods Engrg*, vol. 60, pp. 153–212, 2004.
40. ——, "An overview of variational integrators," in *Finite Element Methods: 1970's and Beyond*, 2004, pp. 98–115.
41. M. West, "Variational integrators," *California Institute of Technology Thesis*, 2004.
42. A. Lew, J. E. Marsden, M. Ortiz, and M. West, "Asynchronous variational integrators," *Arch. Rational Mech. Anal.*, vol. 167, pp. 85–146, 2003.
43. J. E. Marsen and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, pp. 357–514, 2001.

44. B. Anderson and J. Moore, *Linear Optimal Control*. Prentice Hall, Inc, 1971.
45. K. Snyder and T. D. Murphey, "Second-order DMOC using projections," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2010.
46. B. Paynter, "Robodinos: What could possibly go wrong?" *Wired Magazine*, 2009.