

Soft Body Locomotion

Jie Tan *

Greg Turk †

C. Karen Liu ‡

Georgia Institute of Technology

Abstract

We present a physically-based system to simulate and control the locomotion of soft body characters without skeletons. We use the finite element method to simulate the deformation of the soft body, and we instrument a character with muscle fibers to allow it to actively control its shape. To perform locomotion, we use a variety of intuitive controls such as moving a point on the character, specifying the center of mass or the angular momentum, and maintaining balance. These controllers yield an objective function that is passed to our optimization solver, which handles convex quadratic program with linear complementarity constraints. This solver determines the new muscle fiber lengths, and moreover it determines whether each point of contact should remain static, slide, or lift away from the floor. Our system can automatically find an appropriate combination of muscle contractions that enables a soft character to fulfill various locomotion tasks, including walking, jumping, crawling, rolling and balancing.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

Keywords: Soft body simulation, locomotion control, finite element method, optimization.

Links: [DL](#) [PDF](#) [WEB](#) [VIDEO](#)

1 Introduction

In this paper we present a method of animating soft body characters, that is, characters that do not have a skeleton. In particular, our emphasis is on creating animations of soft body creature locomotion, including crawling, walking, rolling and jumping. There are a wide variety of animals in nature that have no skeleton whatsoever. Some examples of such creatures are slugs, starfish, earthworms, octopus, and jellyfish. In addition, many hand-drawn animated characters move in such a flexible manner that they seem to be boneless. The animation principle of *squash-and-stretch* can be seen in its purest form with soft body characters. Finally, as exemplified by our own tongues, even animals with skeletons can have body parts that move without the help of bones. Our research is driven by the intellectual challenge of simulating the locomotion of such soft body creatures, without resorting to any form of rigid elements in our models.

*e-mail: jtan34@gatech.edu

†e-mail: turk@cc.gatech.edu

‡e-mail: karenliu@cc.gatech.edu

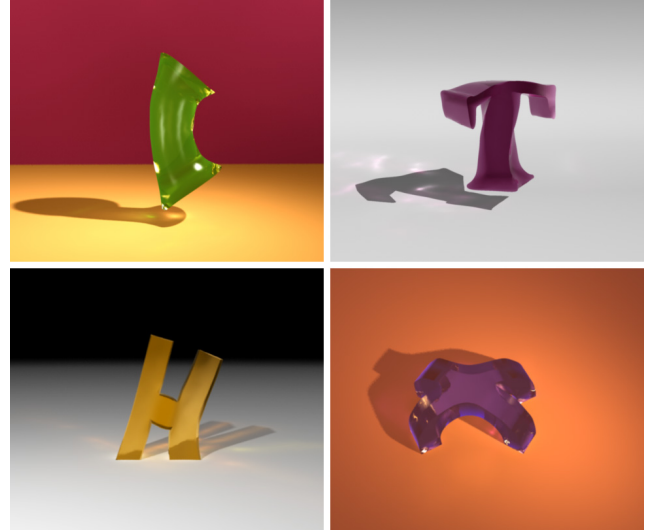


Figure 1: Four alphabetic soft body characters perform different forms of locomotion.

There are two key aspects of anatomy that allow real soft-bodied creatures to move: volume preservation and muscle contractions. Our animation system makes use of these same principles. Soft body tissue is volume preserving, due primarily to the incompressible nature of water. This volume preservation puts constraints on the degree of deformation that a soft body may undergo. We use volume-preserving finite elements to match this aspect of soft body tissue. The second important aspect of soft body creatures is that they control the shape of their body by the contraction of muscles. If such a creature shortens only the muscles that run down the right side of its body, this will cause the creature to bend towards the right. Note that volume preservation and muscle contractions often work in concert to produce motion. If a cylindrical creature uses radial muscle contraction to make itself thinner, then the constraint of volume preservation means that at the same time the creature will stretch lengthwise.

Each of our soft body characters is represented as a tetrahedral mesh and simulated using the finite element method. Our models typically contain hundreds of tetrahedral elements, and controlling such a high degree of freedom model poses a challenge. The aforementioned muscles from real animals provide a way of reducing the degrees of freedom in our characters. In addition to the tet mesh, each of our characters is augmented with a collection of polyline paths, each of which represents a muscle fiber. A character changes its body shape by contracting these muscle fibers, and this induces a shape change in the collection of tetrahedra near the fibers. In theory, such a character could be controlled by specifying the timing of various muscle contractions. However, unlike controlling articulated figures using joint torques, the complex interplay between muscles and soft body shapes makes the control problem exceedingly challenging; even bending a limb of a soft body creature is much more difficult than bending a joint of an articulated figure. For these reasons, we decided that controlling a soft body creature

by specifying the changes to each muscle fiber would be a tedious method of control.

Our system provides a collection of intuitive controls for soft body creature motion, such as moving a point of the character to a given position, or regulating the character's linear or angular momentum. With this collection of intuitive controls, we are able to animate a variety of soft body characters, and in particular, we can demonstrate a wide array of locomotion methods. To move a character, we specify a set of high-level goals (possibly time-varying), and these goals are turned into an objective function that is passed to our solver. For each time step, we formulate and solve a constrained optimization problem, and this gives us new muscle lengths. These muscle lengths induce changes in stress that are applied to the tetrahedral elements, and we then use our physics simulator to advance the system forward in time.

An important part of our constraint solver is contact planning, and this proves to be a challenge for soft bodies. At each time step, our solver must be able to predict how a change to the muscle contractions will influence the points of contact between the character and the ground. For articulated figures, most optimization-based controllers assume that each point of contact is static, which makes contact resolution relatively straightforward to solve. In our system, we cannot assume static contact because sliding and breaking contact turn out to be quite important strategies to control soft characters. For instance, a soft creature may need to widen its base in order to balance, and this means that the points of contact must slide. Depending on the motion goals that are given for a character, the best way to minimize the resulting objective function might be to maintain static contact, to break contact, or to allow sliding contact along the ground. The behavior needs to be decided for each point of contact, and this results in a high dimensional and discontinuous optimization problem. We formulate this as a linear complementarity problem with a quadratic objective function. Although similar problems have been recently proposed in other fields [Braun and Mitchell 2005; Bai et al. 2011], we believe that our solution method is new to graphics.

A natural alternative to our approach would be to represent a character as a rigid, articulated skeleton, and to surround the skeleton with soft tissue that deforms. Such a character representation could even demonstrate elongation and contraction with the use of translational joints. This approach would have several advantages, including the availability of numerous tools that can be used to control an articulated figure. We made a deliberate choice to avoid using rigid elements entirely. We think that using only soft elements will be more likely to result in motions that are more faithful to actual soft body creatures. Our approach avoids the possibility that the character motion shows hints of a hidden skeleton. In addition, using only muscle contractions keeps our character motions "honest" in terms of the magnitude of forces that such characters can apply without a skeleton. Perhaps the most important reason that we avoid the use of rigid elements is our desire to expand the creature body forms that we can simulate using computers. Many animals in nature move without the use of skeletons, so is it possible to create a computer simulation that mirrors this fascinating phenomenon?

The main contributions of our paper are:

1. A new representation of soft body characters that uses muscle fibers to control a volume-preserving finite element mesh. This representation matches the muscle fibers and volume preserving tissue found in real soft body creatures.
2. A framework that provides control of the motion of a soft body character through high-level goals.
3. A new approach for solving the linear complementarity prob-

lem with a quadratic objective function, which enables our controller to utilize a wide variety of contact cases, including static, sliding, rolling and breaking contact.

2 Related Work

Soft body shape control. Controlling physically simulated soft bodies is a practical problem in computer animation. Previous work offers a rich repertoire of techniques that enable the artists to control the shape of soft bodies. Many methods proposed to track a given input animation or keyframes using interpolated resting shapes [Kondo et al. 2005], a constrained Lagrangian solver [Bergou et al. 2007], a linear quadratic regulator [Barbič and Popović 2008], or reduced spacetime optimization [Barbič et al. 2009]. Martin *et al.* [2011] introduced an example-based approach for simulating soft bodies with desired behaviors. The user supplies the system with a few poses to guide the simulation results toward preferred shapes. Shape control for soft bodies has also been applied to physics-based facial animation. Sifakis [2005] formulated an optimization to automatically determine muscle activation that tracks a sparse set of motion capture markers. Our work also aims to track predefined shapes using muscle activation. Unlike Sifakis's work, we do not assume a steady state when computing muscle activation because inertia effects play a key role in the types of motion we focus on in this paper. In addition, while the pose of a face is entirely determined by muscle activation and kinematic parameters, our muscle activation needs to handle issues due to discontinuous contact forces and balance.

Soft body locomotion control. In contrast to shape control, locomotion control for soft bodies is relatively less explored in computer animation. The main difficulty in locomotion is to control an under-actuated system by exploiting external forces. Previous work has shown that mass-spring systems can be used to simulate motion of worms, snakes, and fish [Miller 1988; Tu and Terzopoulos 1994; Grzeszczuk and Terzopoulos 1995]. Miller [1988] utilized anisotropic frictional forces such that a worm can slide forward by contracting elastic body segments. Tu and Terzopoulos [1994] applied a simple fluid dynamic model to provide forward thrust when a fish deforms its body. Recent work by Kim and Pollard [2011a; 2011b] demonstrated that much more complex locomotion can be achieved by effective soft body control. They combined an efficient skeleton-driven FEM simulator and an optimization-based controller to create many interesting behaviors, such as a star fish crawling out of a box and a fish flipping back and forth. Inspired by their compelling results, we wish to generate even more complex locomotion that requires intricate balance control, using only muscle contraction without an actuated skeleton. Consequently, our work demands a more sophisticated contact modeling method during control optimization to handle discrete switches between static contact, sliding contact, and contact breakage.

Contact handling. One broadly applied technique to handle contact is to formulate a linear complementarity problem (LCP). Stewart and Trinkle proposed an LCP formulation using an implicit time-stepping method to guarantee non-penetration, directional friction, and approximated Coulomb's friction cone conditions [Stewart and Trinkle 1996]. Based on the LCP framework, many improved contact models were introduced recently in computer graphics, including using an efficient iterative method [Erleben 2007], a simple staggered sequence of projections [Kaufman et al. 2008], or a progressive constrained manifold refinement [Otaduy et al. 2009]. Velocity-based LCPs for contact modeling can have infinitely many solutions, but general LCP solvers, such as Lemke's algorithm, are incapable of ascertaining the quality of

the solutions for a given criterion. This drawback is particularly undesirable when solving an optimal control problem that exploits the contact and dynamic state of the system. Due to the lack of robust schemes to formulate optimization with arbitrary objective function and linear complementarity constraints, many previous methods explicitly assumed that the contacts remain static [Abe et al. 2007; Jain et al. 2009; Kim and Pollard 2011a] while optimizing control forces subject to equations of motion. This assumption significantly restricts the effectiveness of the controller for locomotion and balance because the controller is not allowed to actively exploit contact breakage, slipping contacts, or rolling contacts to achieve control goals. A few previous studies in mathematics addressed the problems of linear and convex quadratic programs with complementarity constraints (LPCCs and QPCCs) [Hu et al. 2008; Bai et al. 2011]. They showed that global resolution of nonconvex problems in these two subclasses, including those infeasible and unbounded, can be accomplished in finite time. Instead of seeking a general solution, we develop our own specialized QPCC solver for the purpose of contact modeling. We exploit the physical meaning of complementarity constraints as heuristics to greatly improve the solution and performance of the solver.

Soft body simulation. Since the seminal work introduced by Terzopoulos [1987], researchers in computer graphics have simulated a wide variety of deformable phenomena including cloth [Baraff and Witkin 1998; Bridson et al. 2002], elasticity [Müller et al. 2002], and plasticity [O’Brien and Hodgins 1999; Bargteil et al. 2007]. One popular technique is the Finite Element Method (FEM) [Bathe 2007], which uses a tetrahedral or hexahedral discretization to solve dynamic equations. The robustness of FEM simulation can be improved by handling inverted tetrahedra [Irving et al. 2004], remeshing ill-conditioned elements [Bargteil et al. 2007], or preserving volume without locking artifacts [Irving et al. 2007]. To improve the performance of FEM simulation, linear strain model and precomputed stiff matrix are often used. However, these models are only valid for small deformation. To simulate large deformation, Müller *et al.* [2002] proposed a corotational method to fix the volume inflation artifacts. Nesme *et al.* [2005] suggested that linearization around the current deformed configuration reduces ghost torques. Precomputed deformation modes have also been used to interactively deform large structures [James and Pai 2003; Barbič and James 2005; Kim and James 2009].

Muscle modeling. Modeling detailed human musculoskeletal system also requires simulating and controlling soft bodies. Previous work has demonstrated that complex interplay among bones, muscles, ligaments and other soft tissues can be modeled for individual body parts, including the neck [Lee and Terzopoulos 2006], the upper body [Zordan et al. 2006; DiLorenzo et al. 2008; Lee et al. 2009], and hands [Tsang et al. 2005; Sueda et al. 2008]. Using the volumetric data from the visible human data set, Teran *et al.* integrated a B-spline representation for muscles, a tetrahedra mesh for soft tissues, and a triangulated surface for each bone to simulate musculoskeletal behaviors [Teran et al. 2003; Teran et al. 2005]. A striking difference of our work is that we focus on controlling deformation behaviors without skeletal support. This type of control mechanism resembles biomechanical movements using muscular-hydrostats, such as the tentacles of cephalopod mollusks or the trunks of elephants [Kier 1985]. By using muscle contraction alone, we can generate functional motor skills, including elongating, shortening, bending, and twisting. We show that visually appealing behaviors that cannot be produced by skeleton-based systems emerge with appropriate control.

3 System Overview

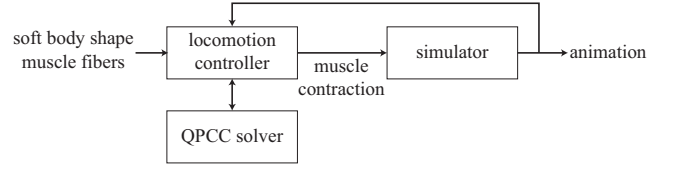


Figure 2: Overview of our system.

We design a wide variety of locomotion controllers for soft bodies, including balance, walking, crawling, jumping, sliding and rolling. Given the geometry of a soft-body creature and the arrangement of its muscle fibers, our controller computes required muscle contraction to propel the creature to achieve desired locomotion while maintaining balance. At each time step, the controller formulates a quadratic program with complementarity constraints (QPCC) to solve for the optimal muscle contraction under discretized dynamic equations of motion and frictional contact constraints. The objective function, assuming a convex quadratic form, can be designed arbitrarily to address control goals of the desired locomotion. The optimal muscle contraction is passed to a FEM simulator to calculate the next state. Figure 2 illustrates the main components of our system.

4 Soft Body Simulation and Modeling

Before we introduce the control algorithms for locomotion, we first describe the methods for simulating soft bodies and computing muscle forces.

4.1 Finite Element Simulation

A soft body creature is represented as a tetrahedral mesh and is simulated using a modified corotational linear FEM [Müller et al. 2002]. We chose FEM instead of a mass-spring model because it is difficult to enforce volume preservation for the material with mass-spring systems. At each time step, the state of the creature, \mathbf{p} , is computed through numerical integration of the dynamic equations of motion:

$$\mathbf{M}\ddot{\mathbf{p}} = \mathbf{f}_x + \mathbf{f}_e + \mathbf{f}_d + \mathbf{f}_m \quad (1)$$

where \mathbf{M} is the mass matrix of the discretized soft body and \mathbf{p} is the nodal position of the deformed shape. The forces on the right hand side, \mathbf{f}_x , \mathbf{f}_e , \mathbf{f}_d , and \mathbf{f}_m , indicate external, elastic, damping, and muscle forces respectively. The external force \mathbf{f}_x includes gravity, contact force, and user perturbation force.

As notation, when we are specifying a quantity \mathbf{q} for a single element, we will write this as $\hat{\mathbf{q}}$. To compute the elastic force for each element, we adapted the method suggested by Nesme *et al.* [2005]:

$$\hat{\mathbf{f}}_e = -\hat{\mathbf{B}}^T \hat{\mathbf{D}} \hat{\mathbf{B}}(\hat{\mathbf{p}} - \hat{\mathbf{R}}\hat{\mathbf{x}}) \quad (2)$$

where $\hat{\mathbf{x}}$ indicates the nodal position in the rest shape and $\hat{\mathbf{R}}$ transforms the element from the reference coordinates to the deformed coordinates. $\hat{\mathbf{B}}$ is the strain-displacement matrix in the deformed coordinates and $\hat{\mathbf{D}}$ is the stress-strain matrix. We use the Poisson ratio 0.45 in $\hat{\mathbf{D}}$ to make the soft body nearly incompressible while avoiding locking artifacts [Irving et al. 2007]. Although volume preservation is not enforced strictly, our experiments show that the volume change is below 15% and is not visually noticeable. This formulation linearizes the elastic force around the current deformed shape, rather than around the rest shape, as used in most

FEM implementations. We chose this formulation because it eliminates “ghost torques” caused by the error of linearization around the rest shape [Nesme et al. 2005]. When the material is soft or when the deformation is small, ghost torques do not cause visible artifacts. However, this formulation is necessary for our case, because soft body locomotion requires large deformation with relatively stiff materials to support the weight of the creature.

We assemble the individual stiffness matrices $\hat{\mathbf{B}}^T \hat{\mathbf{D}} \hat{\mathbf{B}}$ for each element into a large stiffness matrix \mathbf{K} for the whole system. The elastic force for all the FEM nodes can be expressed by $\mathbf{f}_e = -\mathbf{K}(\mathbf{p} - \mathbf{R}\mathbf{x})$. For damping force, we use simple Rayleigh damping model to compute its effect: $\mathbf{f}_d = -\mathbf{C}\dot{\mathbf{p}} = -(\mu\mathbf{M} + \lambda\mathbf{K})\dot{\mathbf{p}}$. We set the damping coefficients $\mu = 0$ and $\lambda = 0.2$ such that the system is slightly over-damped.

4.2 Muscle Modeling

We model muscle fibers as polygonal curves with a small number of segments. Each muscle segment can contract along its current direction, but it cannot extend or bend. Based on the arrangement of muscle fibers, we can bundle them into muscle groups. There are three types of muscles which lead to different control tasks. The *longitudinal muscles* are linear muscles that extend from one end of the body to the other end. Their main function is to shorten or bend the body. The *radial muscles* are a set of short muscles that span a cross-section of the body. When radial muscles contract, the volume-preserving nature of the tissue causes the body to elongate. *Helical muscles* wrap around the body in a helical shape. When a helical muscle contracts, the body twists.

Muscle contraction induces muscle force on nearby FEM elements. Each muscle segment is modeled as a spring with a changeable desired length. The spring force caused by each segment is computed as: $f = k(l_d - l)$, where k is the stiffness of the muscle fiber, l_d is the current desired length, and l is the current length of the segment. We treat f as a virtual force, which is realized by muscle stress imposed on nearby elements. When a muscle segment contracts with the virtual force f in the direction of \mathbf{d} in the reference coordinates, the effect of contraction is as muscle stress \mathbf{F} :

$$\mathbf{F} = \mathbf{U} \begin{bmatrix} f & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{U}^T \quad (3)$$

where \mathbf{U} is the matrix that rotates vector $(1, 0, 0)^T$ in the reference coordinates to be aligned with \mathbf{d} .

Each FEM element may be affected by multiple muscles. The accumulated muscle stress experienced by an element i is a weighted sum of all the muscle stresses that have an influence on the element i , denoted in the deformed coordinates of element i as:

$$\hat{\sigma}_m^i = \sum_j w_{ij} \hat{\mathbf{R}} \mathbf{F}_j \hat{\mathbf{R}}^T \quad (4)$$

where w_{ij} weighs the influence of muscle fiber j on the element i . The value of w_{ij} is based on the shortest distance d_{ij} , from the muscle fiber to the center of the element in the reference coordinates. We use a Gaussian kernel as the attenuation function, $h(d) = \exp(-\frac{d^2}{\sigma^2})$, where σ is the variance of the Gaussian function. The influence weight w_{ij} is defined as

$$w_{ij} = \frac{h(d_{ij})}{\sum_{k \in \text{group}(j)} h(d_{ik})} \quad (5)$$

The denominator in Equation 5 normalizes the influence of muscle fibers within the same group. This normalization allows different

muscle groups, typically with different functionalities, to exert their influence on the soft body simultaneously.

Once we compute the muscle stress for each element $\hat{\sigma}_m$, we calculate the force at each face of the element by multiplying $\hat{\sigma}_m$ with the area-weighted face normal in the deformed coordinates. Finally, we evenly distribute the force at each face to the vertices to obtain \mathbf{f}_m at each node. As a shorthand, we define a muscle force matrix \mathbf{A} to express the relation between the muscle force on each node and the effect of muscle contraction.

$$\mathbf{f}_m = \mathbf{A}(\mathbf{l}_d - \mathbf{l}) \quad (6)$$

Note that $\mathbf{A} \in \mathbb{R}^{3n \times m}$ where n is the number of nodes in the FEM mesh and m is the number of muscle segments, which is also the dimension of our control variables.

4.3 Numerical Integration

To ensure the stability of our system with large time steps, we use an implicit integrator to solve the dynamic equations. After substituting each force terms into Equation 1, we arrive at the following dynamic equation:

$$\mathbf{M}\ddot{\mathbf{p}} = \mathbf{f}_x - \mathbf{K}(\mathbf{p} - \mathbf{R}\mathbf{x}) - \mathbf{C}\dot{\mathbf{p}} + \mathbf{A}(\mathbf{l}_d - \mathbf{l}) \quad (7)$$

Applying the implicit integrator, we can rewrite the equation as,

$$\begin{aligned} \widetilde{\mathbf{M}}\dot{\mathbf{p}}^{n+1} &= \mathbf{M}\dot{\mathbf{p}}^n + \Delta t(\mathbf{f}_x^n - \mathbf{K}(\mathbf{p}^n - \mathbf{R}\mathbf{x}^n) + \mathbf{A}(\mathbf{l}_d - \mathbf{l}^n)) \\ &= \widetilde{\mathbf{f}}^n + \mathbf{f}_c + \widetilde{\mathbf{A}}\mathbf{l}_d \end{aligned} \quad (8)$$

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \Delta t\dot{\mathbf{p}}^{n+1} \quad (9)$$

where superscript n indicates the discretized time index and Δt is the time step. We define $\widetilde{\mathbf{M}} = \mathbf{M} + \Delta t\mathbf{C} + \Delta t^2\mathbf{K}$ and $\widetilde{\mathbf{A}} = \Delta t\mathbf{A}$, and single out the contact force as \mathbf{f}_c , which is part of \mathbf{f}_x and will be discussed in the next section. $\widetilde{\mathbf{f}}^n$ accounts for the remaining terms on the right hand side of Equation 8.

5 Locomotion Control

To create functional locomotion using the simulation framework described in previous section, we need a control algorithm to compute the appropriate muscle contractions. Our control algorithm formulates an optimization at each time step to solve for the desired muscle contraction \mathbf{l}_d that achieves the control goals subject to physical constraints.

5.1 Optimization

We express the objective function in the optimization as a convex quadratic function of the next state of the soft body: $G(\mathbf{p}^{n+1})$. This general function form is sufficient to encode a wide variety of control goals while retaining convexity of the optimization. Using Equation 8 and 9, the optimization minimizes a reparameterized objective function which implicitly enforces the equations of motion:

$$\min_{\mathbf{l}_d} G(\mathbf{p}^n + \Delta t\widetilde{\mathbf{M}}^{-1}(\widetilde{\mathbf{f}}^n + \mathbf{f}_c + \widetilde{\mathbf{A}}\mathbf{l}_d)) \quad (10)$$

In addition to the objective function, the optimization must satisfy two constraints. The first constraint enforces the range of muscle contractions: $0.5\mathbf{l}_0 \leq \mathbf{l}_d \leq \mathbf{l}_0$, where \mathbf{l}_0 denotes the muscle length at the rest pose. The second constraint enforces valid contact under Coulomb’s friction model. We adapt an implicit time-stepping LCP method to regulate contact velocity and contact force, $\mathbf{f}_c = \mathbf{N}f_{\perp} + \mathbf{D}\mathbf{f}_{\parallel}$, where \mathbf{N} is the unit normal vector, \mathbf{D} is a set of tangential

directions at the contact point, and f_\perp and f_\parallel are the magnitudes of normal and tangent forces. The optimization with constraints can be written as

$$\begin{aligned} \min_{l_d, f_\perp, f_\parallel, \lambda} \quad & G(l_d, f_\perp, f_\parallel) \\ \text{subject to} \quad & 0.5l_0 \leq l_d \leq l_0 \\ & \mathbf{0} \leq \begin{bmatrix} f_\perp \\ f_\parallel \\ \lambda \end{bmatrix} \perp \begin{bmatrix} \mathbf{N}^T \mathbf{p}^{n+1} \\ \mathbf{D}^T \mathbf{p}^{n+1} + \mathbf{E}\lambda \\ \mu f_\perp - \mathbf{E}^T f_\parallel \end{bmatrix} \geq \mathbf{0} \end{aligned} \quad (11)$$

where μ is the friction coefficient and \mathbf{E} is a block-diagonal matrix of \mathbf{e} , which is a vector of ones. The complementarity constraints also introduce auxiliary variables λ . The physical meaning of λ is related to the tangent velocity of a sliding contact. Please see Anitescu and Portra [1997] for a complete review of LCP formulation.

A quadratic program with linear complementarity constraints (QPCC) is well known for its nonconvexity and disjunctive features, which cannot be solved efficiently by standard nonconvex solvers. Previous work simplified this problem by assuming that the current contacts will remain static (the velocities of contact points remain zero) at the end of time step. If this assumption is not consistent with the simulated result, the controller will try to correct it at the next time step. In soft body control, assuming static contacts is too restrictive and significantly reduces the effectiveness of the controller. As a result, we cannot drop the complementarity constraints in the optimization. We will introduce a new iterative solver to QPCC for contact modeling in Section 6.

5.2 Low-level Controllers

We develop three types of low-level control mechanisms by formulating different objective functions $G(l_d, f_\perp, f_\parallel)$ in Equation 11. In Section 7, we demonstrate that these three basic mechanisms can be combined to design fundamentally different locomotion controllers.

Momentum control. Regulating momentum is of paramount importance for biped balance and locomotion. Previous work [Machietto et al. 2009] has demonstrated that controlling the linear momentum relative to the contact support is a simple but very effective balance strategy. We use the following objective function to regulate the linear momentum, \mathbf{L} .

$$G(l_d, f_\perp, f_\parallel) = \|\dot{\mathbf{L}}(\mathbf{p}^{n+1}, \mathbf{p}^n) - \bar{\mathbf{L}}\|^2 \quad (12)$$

The desired change of linear momentum $\bar{\mathbf{L}}$ is defined as

$$\bar{\mathbf{L}} = mK_p(\bar{\mathbf{c}} - \mathbf{c}^n) - K_d\mathbf{L}^n \quad (13)$$

where m is the mass of the creature and \mathbf{c} is the center of mass (COM) position. K_p and K_d are the stiffness and damping coefficients for the feedback control. For balance control, the desired COM position $\bar{\mathbf{c}}$ is computed based on the center of the contact support area.

Angular momentum also plays an important role in balance. For soft body creatures, controlling angular momentum is also essential to rolling motion.

$$G(l_d, f_\perp, f_\parallel) = \|\dot{\mathbf{H}}(\mathbf{p}^{n+1}, \mathbf{p}^n, \mathbf{p}^n) - \bar{\mathbf{H}}\|^2 \quad (14)$$

where $\bar{\mathbf{H}}$ denotes the target value for the change of angular momentum and $\dot{\mathbf{H}}$ computes the change of angular momentum at the next state.

Base control. In addition to controlling the momentum, we can increase the contact area to provide a wider range of support to the COM. This balance strategy is particularly interesting for soft bodies. By squashing and stretching its entire body, a soft body creature can adjust its base area at will to maintain balance. We define an objective function that controls the projected base area A by matching its change rate to a desired rate \bar{A} :

$$G(l_d, f_\perp, f_\parallel) = \|\dot{A}(\mathbf{p}^{n+1}, \mathbf{p}^n) - \bar{A}\|^2 \quad (15)$$

We compute A by projecting a defined base area to the ground surface with normal vector \mathbf{n} :

$$A = \frac{1}{2} \sum_i ((b_i - a_i) \times (c_i - a_i))^T \mathbf{n} \quad (16)$$

where index i loops over all triangles in the base area, and a_i , b_i , and c_i are the vertices of the i th triangle. When computing \dot{A} , we evaluate the velocity terms at \mathbf{p}^{n+1} and the position terms at \mathbf{p}^n . This approximation has negligible effect on accuracy, but keeps our objective function convex.

Position and velocity tracking. Direct control of a Cartesian position or velocity is also an effective way to regulate locomotion. For example, tracking the trajectory of a foot is essential for producing a walking gait. The following objective function minimizes the distance between a particular body point at the next time step and a target Cartesian point $\bar{\mathbf{p}}$

$$G(l_d, f_\perp, f_\parallel) = \|f(\mathbf{p}^{n+1}) - \bar{\mathbf{p}}\|^2 \quad (17)$$

where f is a function that selects a node from \mathbf{p}^{n+1} . If we redefine f as a function that computes the COM, Equation 17 can be used to track the COM. Likewise, we can track the relative position of two body points by replacing $f(\mathbf{p}^{n+1})$ with $f_1(\mathbf{p}^{n+1}) - f_2(\mathbf{p}^{n+1})$, where f_1 selects the first node and f_2 selects the second node from \mathbf{p}^{n+1} . We can also use a similar objective function to track velocity.

$$G(l_d, f_\perp, f_\parallel) = \|\dot{f}(\mathbf{p}^{n+1}) - \bar{\dot{\mathbf{p}}}\|^2 \quad (18)$$

6 QPCC for contact modeling

The control framework described in Section 5 requires an efficient QPCC solver that can handle 50 to 100 complementarity variables. Solving QPCC in general is difficult due to the presence of the linear complementarity constraints. A naïve way to solve QPCC is to evaluate all the valid combinations of the complementarity constraints and output the minimizer. This exhaustive method is guaranteed to find the global minimum. However, the computational time grows exponentially with the number of variables involved in the complementarity constraint. In our control problem, we have 10 variables for each contact point (one for f_\perp , eight for f_\parallel and one for λ). Thus, a few contact points alone will render the exhaustive method computational impractical.

We propose a more efficient way to solve a QPCC for contact problems, such as Equation 11. Our iterative QPCC solver starts with an initial guess, which is a set of linear constraints that are compatible with the complementarity conditions. For the initial guess, we manually set some elements of f_\perp , f_\parallel , and λ to be zero and their complementary pairs to be nonnegative, in a way that the state of those variables has a physical meaning. For example, if we assume

all contact points are static, we arrive at the following convex QP:

$$\min_{\mathbf{l}_d, \mathbf{f}_\perp, \mathbf{f}_\parallel, \lambda} G(\mathbf{l}_d, \mathbf{f}_\perp, \mathbf{f}_\parallel) \quad (19)$$

subject to

$$0.5\mathbf{l}_0 \leq \mathbf{l}_d \leq \mathbf{l}_0$$

$$\mathbf{0} \leq \mathbf{f}_\perp, \quad \mathbf{N}^T \dot{\mathbf{p}}^{n+1} = \mathbf{0} \quad (20)$$

$$\mathbf{0} \leq \mathbf{f}_\parallel, \quad \mathbf{D}^T \dot{\mathbf{p}}^{n+1} + \mathbf{E}\lambda = \mathbf{0} \quad (21)$$

$$\mathbf{0} = \lambda, \quad \mu \mathbf{f}_\perp - \mathbf{E}^T \mathbf{f}_\parallel \geq \mathbf{0} \quad (22)$$

After solving the above QP, we examine the complementarity conditions at the minimizer. We identify those inequality constraints that reach their boundary at the minimizer as candidates for pivoting. Our algorithm pivots one of those candidates at a time. That is, we set the candidate to equality constraint and flip its complementary counterpart from equality to inequality. By pivoting the complementarity constraints, we formulate a new QP with a set of different linear constraints and we solve for the minimizer for this new QP. We repeat this process until all the candidates reach the local minimum of the QPCC, i.e. until we encounter a minimizer that lies in the interior of the feasible region. The candidate that yields the best local minimum is returned as the solution of the QPCC. Our QPCC solver explores the nonconvex feasible region based on the following heuristics: Each pair of the complementarity constraints defines a feasible region formed by two intersecting half-hyperplanes. If a minimizer hits the boundary of the half-hyperplane, exploring the other half-hyperplane might give a better minimizer. Figure 3 shows a two dimensional example.

Our algorithm further exploits the structure of a contact problem to improve the performance. Instead of arbitrarily selecting a candidate to pivot, we can group the complementarity constraints according to their physical meaning and pivot a whole group together. There are three different situations for each contact point: static, sliding and contact breakage. Pivoting constraints in Equation 20 indicates a switch between a static (or a sliding) contact and contact breakage. Pivoting constraints in Equation 21 and 22 indicates a switch between static and sliding contact. For example, if $\mathbf{f}_\perp(i) = 0$ is the result of solving the QP (Equation 19), it implies that breaking i th contact point might lead to a better minimizer for the QPCC. The solver will pivot the corresponding constraints: $\mathbf{f}_\perp(i) \geq 0 \rightarrow \mathbf{f}_\perp(i) = 0$ and $(\mathbf{N}^T \dot{\mathbf{p}}^{n+1})(i) = 0 \rightarrow (\mathbf{N}^T \dot{\mathbf{p}}^{n+1})(i) \geq 0$. The new QP will be solved subsequently.

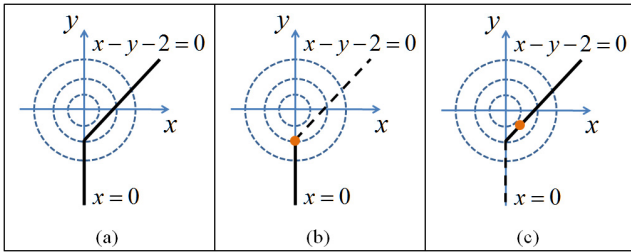


Figure 3: A simple 2D QPCC example. The complementarity constraints are $0 \leq x \perp x - y - 2 \geq 0$. (a) The feasible region lies in two intersecting half-hyperplanes, shown as two black line segments. (b) With the initial guess of $x = 0$ and $x - y - 2 \geq 0$, the minimizer, shown as an orange dot, is located at the boundary of the inequality constraint. (c) After pivoting the constraint, setting $x \geq 0$ and $x - y - 2 = 0$, we find a better minimizer (global minimizer in this simple case).

Conversely, when a free point restores a static contact, we apply the opposite pivoting.

If a friction cone condition (Equation 22) for the i th contact point needs to be pivoted, this implies that the i th contact point is about to slide and switching it from static to sliding might lead to a better minimizer. The solver then changes the inequality constraint $(\mu \mathbf{f}_\perp - \mathbf{E}^T \mathbf{f}_\parallel)(i) \geq 0$ to equality and changes the corresponding equality constraint $\lambda(i) = 0$ to inequality. In addition, we need to pivot some constraints in Equation 21 to specify the direction of the sliding contact, which can be estimated using the static friction from the current minimizer. We project this static friction force to each of the tangential direction of the i th contact point $\mathbf{D}(i)$ and find the two directions (the m th and n th direction in $\mathbf{D}(i)$) that have the largest magnitude. The sliding force direction is estimated to be along the convex combination of m th and n th directions. We pivot the constraints in the following way,

$$\mathbf{f}_\parallel(i, m) \geq 0, \quad (\mathbf{D}^T \dot{\mathbf{p}}^{n+1} + \mathbf{E}\lambda)(i, m) = 0$$

$$\mathbf{f}_\parallel(i, n) \geq 0, \quad (\mathbf{D}^T \dot{\mathbf{p}}^{n+1} + \mathbf{E}\lambda)(i, n) = 0$$

$$\mathbf{f}_\parallel(i, j) = 0, \quad (\mathbf{D}^T \dot{\mathbf{p}}^{n+1} + \mathbf{E}\lambda)(i, j) \geq 0, \quad \forall j \neq m, n$$

where $\mathbf{f}_\parallel(i, j)$ is the magnitude of the friction force along the j th direction for the i th contact point¹. For the special case, where the friction force is exactly along the m th (or n th) direction, we only pivot the complementarity constraints involving the m th (or n th) direction. For a switch between sliding to static contact, we use the same pivoting mechanism but pivot the constraints the opposite way.

Instead of searching exhaustively in the feasible region of the QPCC, our solver systematically explores the feasible region based on the above mentioned heuristics. Although the objective value is not guaranteed to decrease monotonically, our experiments show that the objective value decreases drastically within a small number of iterations. The minimizer found by our solver is, in all the experiments, significantly closer to optimal than the one solved under the static contact assumption. We report the results of the experiments in Section 8.

Implementation. Our solver requires a feasible initial guess. We can use constraints under static contact assumption (Equations 20-22) as an initial guess, or the solution from the previous QPCC when it is available (warm start). Occasionally, both initialization methods fail to generate feasible solutions. In that case, we assume the same muscle activation as in last time step, remove the objective function and muscle length constraints from Equation 11 and solve a pure LCP. The contact situation from the LCP solution is then used as the initial guess for the QPCC.

We implement the QPCC solver using a graph expansion algorithm. Each QP with linear constraints is a node in the graph. We visit each node twice, starting from the initial guess as the root. In the first visit, we solve the QP, assign the objective value to the node, and store the set of candidates to be pivoted. After first visit, we push the node into a priority queue based on its objective function value. A node is visited the second time when it is at the top of the queue. In the second visit, we pivot the constraints from its candidate set. Each pivot generates a child node. We discard the child node if it already exists in the current graph. If the child node is new, we visit the node for the first time and push it into the queue. The second visit is completed when all the constraints in the candidate set are pivoted. We then pop the next node in the queue and repeat

¹ $\mathbf{f}_\parallel(i, j)$ is actually the $(i \cdot N + j)$ th element of \mathbf{f}_\parallel assuming that N tangential directions are used for the linearized friction cone for each contact point.

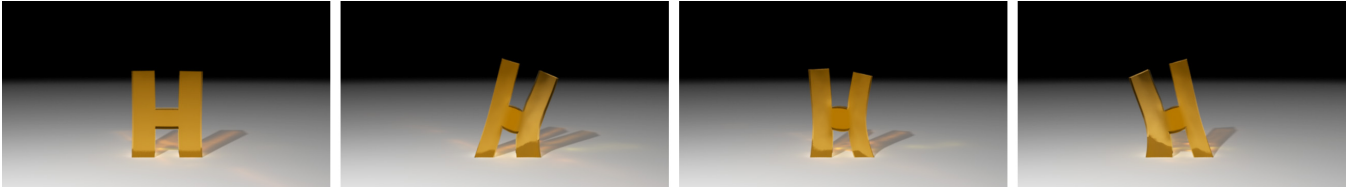


Figure 4: An H-shaped soft body character does its morning exercises by swinging its body from one side to the other.

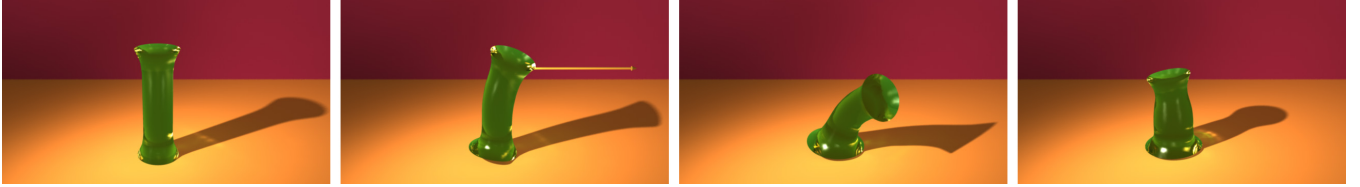


Figure 5: An I-shaped soft body character tries to maintain balance under perturbation by regulating its momenta, widening its base and lowering its center of mass.

this process. The algorithm terminates when the priority queue is empty or the number of visited nodes exceeds a threshold. The final solution is the best minimizer found so far by the QPCC solver.

7 Results

In this section we describe the results of our soft body locomotion controllers. Please see the accompanying video to watch the locomotion animations. Our system is implemented in C++, and we generated the tetrahedral mesh for FEM simulation using TETGEN [Si 2006]. We used the GPU to create layered depth images for collision detection, and we used contact patches (multi-resolution volume contact) [Allard et al. 2010] instead of points as the contact primitives. For each contact patch, we use eight tangential directions to linearize the friction cone, which provides sufficient accuracy while keeping the QPCC tractable. The examples were run on a workstation with a 2.26GHz CPU and 4GB of memory. All the data of our locomotion examples are summarized in Table 1.

We design many different shapes of the soft body characters, all of which are chosen from the English alphabet. Figure 4 shows an H-shaped character doing morning exercises. The character is designed with four longitudinal muscles and one radial muscle for each leg (Figure 10a). It is animated by specifying the trajectory of the desired center of mass (COM), which is moved left/right by a sine function. We use the position and velocity tracking con-

troller from Section 5.2 to track the desired COM. Note that when the “H” swings left, its right side elongates and gets thinner while the left side shortens and becomes fatter due to the volume preservation. This animation clearly exemplifies the principle of *squash and stretch*.

Balance. We design an I-shaped character (Figure 5) to demonstrate static balance. We gave the character four longitudinal muscles that allow it to bend in any direction. This character is perturbed by a large force exerting at its head, and it attempts to recover its balance. Static balance of the “I” turns out to be one of the most difficult task among all our examples. The geometry of the letter “I” does not have limbs or other appendages to help it regulate the linear and angular momentum. The squishy body and lack of skeletal support make the task even more challenging. In addition to momentum control for balance, which is not enough to prevent the “I” from falling, we exploit the advantage of its flexible body shape. We include a term in the objective function that encourages it to widen its support base. With this wider base, the contact area is increased and the COM is lowered, which helps with the balance task. Without our QPCC solver, base widening would be difficult to achieve, because it requires frequent switching from static to sliding contacts. In addition, we observe that right after the perturbation, half of the base is lifted from the ground and the contact area concentrates on the rim of the base to provide the maximum amount of torque to combat the perturbation. It is similar to a human lifting his or her heels and using only toes to balance when pushed from behind. This natural contact strategy emerges automatically from our QPCC solution. To compare our QPCC solver and a more commonly used QP solver with linear constraints, we produced two animation sequences of the “I” balancing, one with each solver. QPCC produced natural and effective balance motions, including changing contact situation, lowering the COM, widening the base and regulating momentum. In contrast, the QP solver (which only allows for static contact constraints) resulted in a falling motion.

Sliding. In the example of Figure 6, instead of applying a perturbation force that lasts for a short time, we exert an continuously increasing pulling force on the “F” standing on a slippery surface. We design a sliding balance controller for this special balance task. The controller estimates the optimal relative position between the center of base (COB) and the COM such that the total angular momentum is zero. We use the position and tracking controller to track

examples	# tets	# dofs	# contact patches	sim time	opt time	total time
exercise(H)	1901	52	4	0.54	0.05	0.59
balance(I)	1066	48	4	0.31	0.28	0.59
slide(F)	705	48	4	0.25	0.23	0.48
jump(I)	1066	104	4	0.33	0.48	0.81
jump(T)	1219	51	4	0.63	0.30	0.93
roll(O)	911	40	6	0.26	0.18	0.44
crawl(I)	620	26	8	0.18	0.24	0.42
walk(X)	1128	112	4	0.54	0.70	1.24

Table 1: Parameters and performance of examples. # tets: the number of elements in the FEM simulation. # dofs: the number of muscle degrees of freedom for the soft body. Sim time, opt time and total time are the average simulation, optimization and total time (in second) per frame.

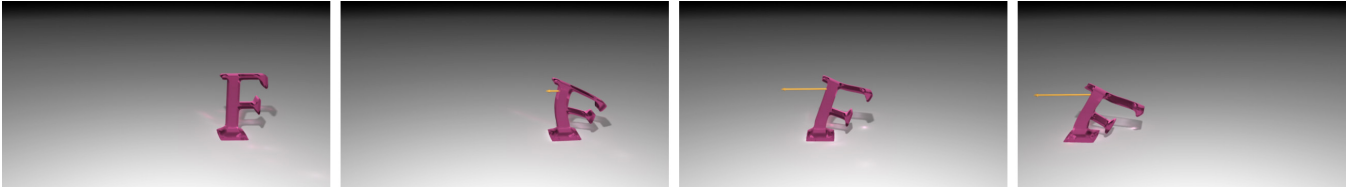


Figure 6: An F-shaped soft body character maintains balance under a persistent and continuously increasing pulling force on a slippery surface. It actively leans backward to avoid tipping over.

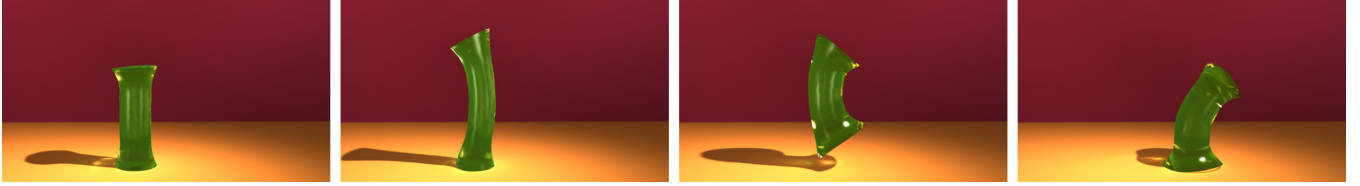
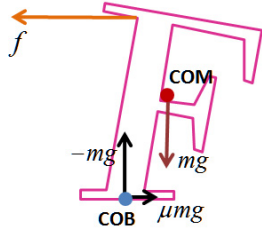


Figure 7: An I-shaped soft body character squashes and stretches its whole body to jump forward.

the optimal COB.

The sliding balance controller also benefits from our QPCC solution since planning the movement of the COB involves planning the change of contact situation (from static to sliding). We instrument the vertical stroke of the “F” with four longitudinal muscles. Even though no muscle resides in the horizontal parts of this character, the two horizontal strokes are still influenced by the muscles in the main body. The first sequence of sliding balance in the video shows that the “F” leans left while it is dragged towards the right. As the drag force increases, the “F” leans more and more to prevent from tipping over. The second sequence shows the sliding motion when it is dragged to the left. The sliding motion is different from the first one due to the asymmetry of the body shape. The elongation and oscillation of the top stroke of the “F” demonstrates the animation principle of *follow through*. In the third sequence, we applied the sliding balance controller 0.3 second after the start of dragging to delay the character’s response time. The slow response of the “F” makes it difficult to maintain the optimal COB-COM relative position. It struggles to keep balance by constantly switching between sliding and breaking contact (small jumps), and eventually it manages to balance. These changes of contact, due to the QPCC solver, makes the controller more robust and the soft body character more lifelike.



Jumping. Jumping is an visually interesting form of locomotion for soft body characters as it is often seen in cartoons and animations. Our jumping controller consists of three separate controllers for takeoff phase, airborne phase, and landing phase. During the takeoff phase, we use the position and velocity tracking controller to follow a desired trajectory of the COM. We also set $\dot{\bar{H}} = \mathbf{0}$ to the angular momentum controller, which prevents large rotation at takeoff. During the airborne phase, we control the relative position between the COB and COM. Extending the COB towards the direction of jumping helps the character balance after landing. Upon landing, we switch to the static balance controller. Figure 7 shows a forward jumping motion of the same character “I” with a slightly different fiber arrangement. We add four more longitudinal muscles

and one radial muscle to help it with this highly dynamic motion. Another sequence in the video shows successive jumps in place. Figure 8 demonstrates a twist jump and the use of helical muscles (Figure 10b). Before the “T” takes off, we set $\dot{\bar{H}} = (0, 600, 0)^T$ to make it twist its body.

Rolling. In the accompanying video, we also demonstrate locomotion by rolling. We designed an O-shaped character with two loops of muscle fibers arranged as two concentric circles (Figure 10c). Each fiber consists of 20 independent segments, which allow the “O” to control its shape locally. The rolling motion is initiated by moving the COM in front of the contact patches. We use the angular momentum control to make it roll. In the first animation, we set the desired change of angular momentum $\dot{\bar{H}}$ to be $(0, 0, -200)^T$ in the first 90 frames. We observe that the character actively changes its shape by shifting its weight to the right in order to roll. After the character starts rolling, we disable the controller

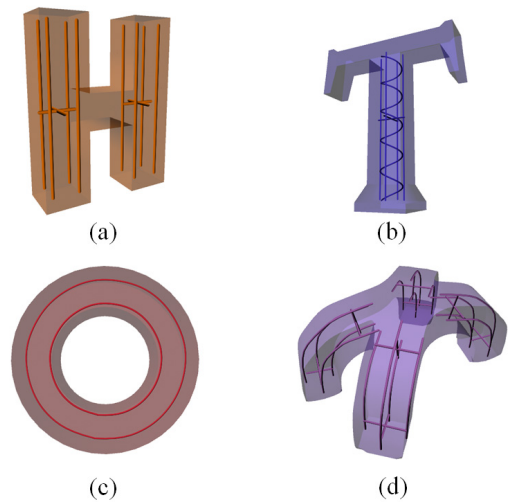


Figure 10: Examples of the muscle fiber designs for various soft body characters. Each curve inside the character represents a muscle fiber, which consists of a number of independently contracting degrees of freedom.



Figure 8: A T-shaped soft body character twists using helical muscles when jumping.

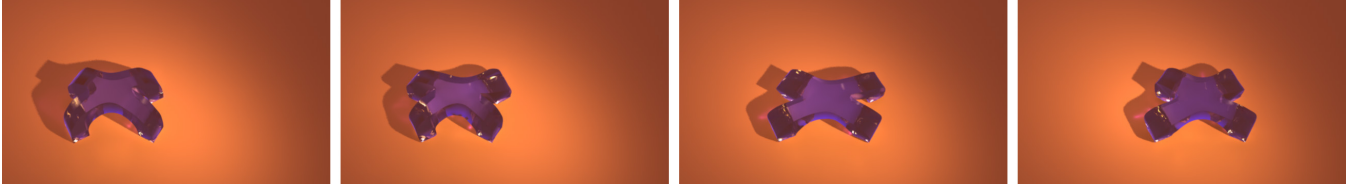


Figure 9: An X-shaped soft body quadruped walks by slowly lifting and moving one foot at a time.

and simulate the passive rolling. The character recovers to its original symmetric rounded shape and the rolling stops after a while due to friction. In the second animation, we compare our result with the motion solved by a QP using the static contact assumption. When we only allow static contact, the “O” never begins rolling because this motion requires the character to break contact, which is prohibited by the static contact assumption (Equation 20). The third sequence shows that the “O” starts to roll right, decelerates, stops and rolls to the left by applying a time varying \dot{H} to the controller.

Crawling. Crawling is often used by soft body creatures in nature, such as earthworms. To demonstrate crawling motion, we flatten the “I” and lay it down on the ground. In addition to the four longitudinal muscles run along four sides of the body, we add another radial muscle in the middle of its body to facilitate the elongation of the body. We specify the trajectory of its four corners for the crawling motion; the back of the character moves while it is contracting, and the front moves when it elongates. We use the position and velocity tracking controllers to match the trajectory. As Miller noted, such creatures have oriented scales that result in anisotropic friction [Miller 1988]. We incorporate just such an anisotropic friction into our contact model by modifying the contact force to $\mathbf{f}_c = \mathbf{N}\mathbf{f}_\perp + \mathbf{D}\mathbf{S}\mathbf{f}_\parallel$, where \mathbf{S} is a diagonal scaling matrix that modulates the frictional force according to the direction of motion. We set the friction coefficient in the backward direction to be 10 times larger than all other directions. In the accompanying video, we demonstrate the earthworm style of crawling. The whole body of the character lies flat on the ground at all times and it moves forward by repeatedly shortening and elongating its body. The contact strategy of this form of crawling is complex. During shortening, the front end of the body is in static contact while the rear end is sliding forward. During elongating, the front end switches to sliding contact while the rear end switches to static contact. It is challenge to capture this complex contact strategy using the traditional control mechanism, but it emerges automatically by solving QPCC.

We also demonstrates an inchworm style of crawling, using the same body geometry and muscles as the earthworm. For this style of motion, the body bends upward periodically at the middle and the contacts mostly concentrate at the two ends of the body. We achieve this effect using the same controller as in the earthworm style crawling with an additional constraint that the upper longitudinal muscle cannot contract. While other muscles contract to tracking the trajectory, the asymmetric muscle contractions bend the body upwards naturally.

Walking. Figure 9 demonstrates the walking motion of an X-shaped quadruped. We instrument four longitudinal muscles and two radial muscles for each limb of the “X” (Figure 10d) and specify the trajectories of four corners on each foot. Using the position and velocity tracking controller, the walking motion emerges from muscle contraction computed by the algorithm. The first sequence in the accompanying video shows a careful and slow gait that moves only one foot at a time. The second walking sequence shows a faster walking gait by simultaneously lifting and moving two feet at a time. The breaking of contact when a foot lifts from the ground is handled automatically by the QPCC solver.

8 Evaluation and Limitations

QPCC. To evaluate our QPCC solver, we tested it on 10 QPCC problems with 98 variables and 40 pairs of linear complementarity constraints. We compared our solutions with the ground truth, as well as with the solutions based on the static contact assumption. The ground truth is computed by an exhaustive search, i.e., solving a QP for every combination of complementarity variables and selecting the one with the lowest objective value. We evaluated our results using a “gap ratio”, defined as the ratio of difference in the optimal value between our solver and the ground truth, to the difference in the optimal value between the static contact assumption and the ground truth. On average of 10 problems, the gap ratio is 6.29, indicating that our solver yields solutions 6.29 times closer to the ground truth than the solutions based static contact assumption. We also selected the best case and the worst case according to the gap ratio and reported them in Table 2.

Limitations. Our system has a few limitations. The optimization scheme described in Section 5 only optimizes the control variables for the next time step. This type of greedy algorithm sometimes leads to unnaturally large muscle contraction or discontinuities in motion. For example, the rolling “O” demo in the accompanying video exhibits some unnatural vibration. Furthermore, the greedy algorithm prevents us from simulating anticipatory behaviors in motion. For example, we were not able to develop a “cartwheel” controller for the I-shaped character because a natural and stable cartwheel motion requires optimizing a long-window of trajectory. This issue can potentially be solved by implementing long-horizon optimization or model predictive control methods.

QPCC is an NP-hard problem [Braun and Mitchell 2005] and our method provides an effective heuristic. Although the empirical re-

	QPCC			QP (Static Contact Assumption)			Ground Truth		
	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst
Objective Value	1483.16	21.79	697.91	5222.84	2072.00	1246.95	772.85	0.00	0.00
Num Iterations	17	3	31	1	1	1	10000	10000	10000

Table 2: The results of the numerical experiments of the QPCC solver.

sults showed that our QPCC solver can effectively solve contact resolution problem and balance between the quality of solution and computational time, the QPCC solver does not guarantee finding the minimizer in polynomial time. In the worst case, it takes the same amount of time as the exhaustive search to find the global minimizer.

9 Conclusion

We have presented a system for animating soft body characters, with a particular emphasis on locomotion. Key aspects of our approach include the coordinated deformation of groups of finite elements using virtual muscle fibers, the specification of high-level goals by the animator, and the use of a new solver that handles static, sliding and breaking contact cases. Our system allows us to create soft body characters that demonstrate a variety of locomotion behaviors, including crawling, hopping, walking, sliding and rolling. Our characters move in an organic manner, and they follow the animation principles of anticipation, squash-and-stretch, and follow through.

One of the issues that we hope to explore in the future is to expand our solution techniques to handle longer-term goals that cannot be reached using our current optimization method. In this work, all muscles are manually designed. We would like to develop an automatic muscle design algorithm to incorporate more sophisticated muscle structures. We would also like to investigate muscle design for chunkier creatures. For example, it is not immediately obvious how muscle fibers should be arranged for the Stanford Bunny. Another possibility is to note that in our current system, we only use contracting muscle fibers in order to change the shape of our characters. It would be interesting to explore other forms of shape control, such as elongating muscles or sheets of virtual muscles. Another possible direction would be to explore the animation of soft body characters in water, since many real soft-body creatures live in an aquatic environment. Finally, our current animator controls are provided as program modules, and an easier way to use them would be to plug them together using a graphical user interface.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. We thank Yuting Gu for modeling, rendering and video editing. We also thank Yuting Ye for her suggestions and all other members of Gatech Graphics Lab for their help on this work. This work was funded by NSF CCF-811485, IIS-11130934 and Alfred P. Sloan Foundation.

References

ABE, Y., DA SILVA, M., AND POPOVIĆ, J. 2007. Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, 249–258.

ALLARD, J., FAURE, F., COURTECUISSÉ, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. 2010. Volume contact con-

straints at arbitrary resolution. *ACM Trans. Graph.* 29 (July), 82:1–82:10.

ANITESCU, M., AND POTRA, F. A. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics* 14, 231–247.

BAI, L., MITCHELL, J. E., AND PANG, J.-S. 2011. On convex quadratic programs with linear complementarity constraints. *In submission*.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, 43–54.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005)* 24, 3 (Aug.), 982–990.

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. *ACM Trans. on Graphics (SIGGRAPH Asia 2008)* 27, 5, 163:1–163:10.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. on Graphics (SIGGRAPH 2009)* 28, 3.

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph.* 26 (July).

BATHE, K.-J. 2007. *Finite Element Procedures*. Prentice-Hall, London.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: Toward Directable Thin Shells. *ACM Transactions on Graphics (SIGGRAPH)* 26, 3 (jul), 50:1–50:10.

BRAUN, S., AND MITCHELL, J. E. 2005. A semidefinite programming heuristic for quadratic programming problems with complementarity constraints. *Computational Optimization and Application* 31, 5–29.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, 594–603.

DILORENZO, P. C., ZORDAN, V. B., AND SANDERS, B. L. 2008. Laughing out loud: control for modeling anatomically inspired laughter using audio. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, 125:1–125:8.

ERLEBEN, K. 2007. Velocity-based shock propagation for multi-body dynamics animation. *ACM Trans. Graph.* 26 (June).

GRZESZCZUK, R., AND TERZOPOULOS, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 63–70.

HU, J., MITCHELL, J. E., PANG, J.-S., BENNETT, K. P., AND KUNAPULI, G. 2008. On the global solution of linear programs

- with linear complementarity constraints. *SIAM Journal on Optimization* 19, 445–471.
- IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, 131–140.
- IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph.* 26 (July).
- JAIN, S., YE, Y., AND LIU, C. K. 2009. Optimization-based interactive motion synthesis. *ACM Transaction on Graphics* 28, 1, 1–10.
- JAMES, D. L., AND PAI, D. K. 2003. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.* 22 (January), 47–82.
- KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.* 27 (December), 164:1–164:11.
- KIER, W. M. 1985. Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats. *Zoological Journal of the Linnean Society* 83, 307–324.
- KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph.* 28 (December), 123:1–123:9.
- KIM, J., AND POLLARD, N. S. 2011. Direct control of simulated non-human characters. *IEEE Computer Graphics and Applications* 31, 4 (July), 56–65.
- KIM, J., AND POLLARD, N. S. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics* 30, 5 (October).
- KONDO, R., KANAI, T., AND ANJO, K.-I. 2005. Directable animation of elastic objects. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, 127–134.
- LEE, S.-H., AND TERZOPOULOS, D. 2006. Heads Up! Biomechanical Modeling and Neuromuscular Control of the Neck. *ACM Transactions on Graphics* 25, 3 (July), 1188–1198.
- LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.* 28 (September), 99:1–99:17.
- MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. R. 2009. Momentum control for balance. *ACM Trans. Graph.* 28 (July), 80:1–80:8.
- MARTIN, S., THOMASZEWSKI, B., GRINSFELD, E., AND GROSS, M. 2011. Example-based elastic materials. *ACM Trans. Graph.* 30 (Aug.), 72:1–72:8.
- MILLER, G. S. P. 1988. The motion dynamics of snakes and worms. *SIGGRAPH Comput. Graph.* 22 (June), 169–173.
- MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, 49–54.
- NESME, M., PAYAN, Y., AND FAURE, F. 2005. Efficient, physically plausible finite elements. In *Eurographics 2005, Short papers, August, 2005*, J. Dingliana and F. Ganovelli, Eds.
- O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, 137–146.
- OTADUY, M. A., TAMSTORF, R., STEINEMANN, D., AND GROSS, M. 2009. Implicit contact handling for deformable objects. *Computer Graphics Forum (Proc. of Eurographics)* 28, 2 (apr).
- SI, H., 2006. Tetgen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator, January.
- SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.* 24 (July), 417–425.
- STEWART, D., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. *International Journal of Numerical Methods in Engineering* 39, 2673–2691.
- SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculo-tendon simulation for hand animation. *ACM Trans. Graph.* 27 (August), 83:1–83:8.
- TERAN, J., BLEMKER, S., HING, V. N. T., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, 68–74.
- TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics* 11 (May), 317–328.
- TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, 205–214.
- TSANG, W., SINGH, K., AND FIUME, E. 2005. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, 319–328.
- TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 43–50.
- ZORDAN, V. B., CELLY, B., CHIU, B., AND DILORENZO, P. C. 2006. Breathe easy: model and control of human respiration for computer animation. *Graph. Models* 68 (March), 113–132.