Discovery of Complex Behaviors through Contact-Invariant Optimization

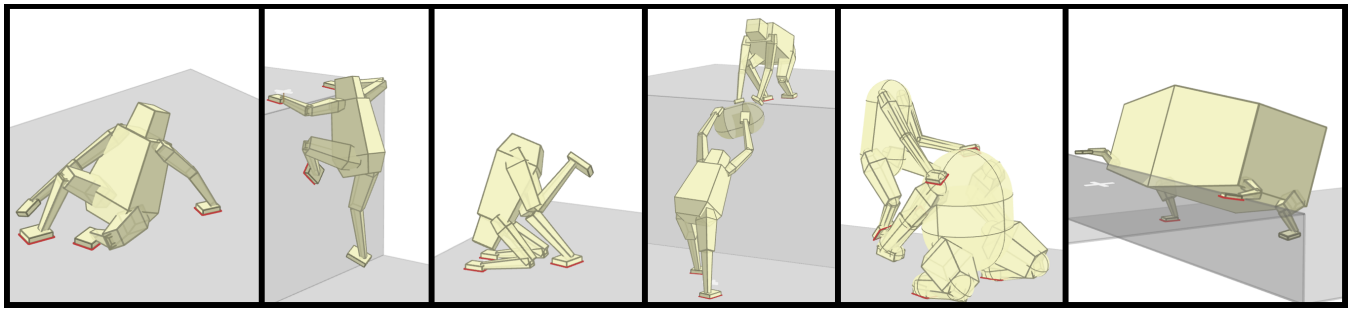

Figure 1: *A selection of motions synthesized by our algorithm.*

Abstract

We present a motion behavior synthesis framework capable of producing a wide variety of important human behaviors that have rarely been studied, including getting up from the ground, crawling, climbing, moving heavy objects, acrobatics (hand-stands in particular), and various cooperative actions involving two characters and their manipulation of the environment. Our framework is not specific to humans, but applies to characters of arbitrary morphology and limb configuration. The approach is fully automatic and does not require domain knowledge specific to each behavior. It also does not require pre-existing examples or motion capture data.

At the core of our framework is the contact-invariant optimization (CIO) method we introduce here. It enables simultaneous optimization of contact and behavior. This is done by augmenting the search space with scalar variables that indicate whether a potential contact should be active in a given phase of the movement. These auxiliary variables affect not only the cost function but also the dynamics (by enabling and disabling contact forces), and are optimized together with the movement trajectory. Additional innovations include a continuation scheme allowing helper forces at the potential contacts rather than the torso, as well as a feature-based model of physics which is particularly well-suited to the CIO framework. We believe that CIO can also be used with a full physics model, but leave that extension for future work.

CR Categories: I.3.1 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: Physics-Based Animation, Control

1 Introduction

Automated synthesis of complex human behaviors is one of the long-standing grand challenges in computer graphics, that would also have an impact on robotics, biomechanics, and movement neuroscience. An automated synthesis method would ideally be capable of creating motions that span the space of all possible human behaviors. In addition, such a method would not require expert or labor-intensive authoring in the form of keyframes, reference trajectories, or any specific details of the intended movement. It would also not require any direct or indirect use of motion capture data. Instead, movement details and complexity should emerge from an automated procedure whose only inputs are intuitive high-level goals that are easy to specify.

The most progress towards this ambitious agenda has been made in the domain of walking. After three decades of intensive research, we now have algorithms that can make simulated humanoids walk robustly and realistically in response to high-level interactive inputs such as desired body velocity and orientation. These algorithms are successful because they exploit domain-specific knowledge: state machines synchronized to the relatively simple and stereotypical pattern of foot-ground contacts, reduced models based on inverted-pendulum dynamics or other features important for walking, and trajectory optimization methods that rely on customized cost functions and manual specification of contacts. The success of these methods comes at the price of limited generality: they provide a unique and different type of model and solution for each of the common locomotion tasks such as walking, running and jumping. With the current state-of-the-art in automated motion synthesis, any additional complex behavior would require a new movement model carefully crafted by experts from scratch. Each of these new movement models would require specific domain knowledge carefully integrated into the motion synthesis algorithm.

This behavior-specific approach to motion synthesis is at odds with the richness and expressiveness of human motor behavior, exhibited in seemingly infinite complex movements that do not fall into standard categories such as locomotion or reaching. Often, these behaviors are more challenging than locomotion in the sense that they involve longer, more complex (spatially and temporally) movement plans, and less stereotypical or cyclic movements. Examples include movements that use more than just legs, or just arms, but that use other parts of the body, movements that can represent the full space of interactions between the human body and the ground and other arbitrary objects in the environment, complex manipulations of objects by one or many humans collaboratively, hand-walking, climbing – to name just a few.

Equally importantly, an automatic framework for synthesis of complex behaviors should not be restricted to human morphology. It should be capable of creating complex behaviors for arbitrary real or invented character morphologies, creating a rich gamut of behaviors that spans all imaginable interactions with the environment, and all imaginable movement goals.

In this paper we present a step towards a more general yet fully automated framework for behavior synthesis, capable of producing a wide variety of less commonly studied but important human behaviors. These include getting up from the ground, crawling, climbing, moving heavy objects, acrobatics (hand-stands in particular), and various cooperative actions involving two characters and their manipulation of the environment. The framework is not specific to humans, but can synthesize behaviors for other imagined morphologies.

1.1 The key idea: Contact-Invariant Optimization (CIO)

As with prior methods for automated behavior synthesis, our CIO method also comes down to exploiting domain-specific knowledge. The important difference is that the domain to which our method is tailored is much larger, and includes any behavior of any articulated character where contact dynamics are essential. This is a very large domain because almost all limb movements performed on land are made for the purpose of establishing contact with some object (including the ground) and exerting forces on it. A suitable set of contacts provides actuation: once you grasp an object you can manipulate it; once you plant your feet on the ground you can generate forces on your torso. Complex movements tend to have phases within which the set of active contacts remains invariant. Such invariance greatly reduces the space of candidate movements. In complex behaviors and in complex environments, however, it is difficult to know in advance what these contact sets should be and how they should change from one phase to the next. Unlike prior work on walking where contact information was specified manually and left outside the scope of numerical optimization, we believe that discovering suitable contact sets should be the central goal of optimization. Once this is done, optimizing the remaining aspects of the movement tends to be relatively straightforward.

Intuitively, CIO is a way of reshaping a highly discontinuous and local-minima-prone search space of movements and contacts, into a slightly larger but much better-behaved and continuous search space that enables optimization strategies to find good solutions. The main technical innovation in the CIO method is the introduction of scalar variables that indicate whether a potential contact should be active in a given phase. These auxiliary variables affect not only the cost function but also the dynamics (by enabling and disabling contact forces), and are optimized together with the movement trajectory. In this way we provide the optimizer with abstract but nevertheless useful information: namely that movements should have phases, and that the contact set should remain invariant within each phase. The specific sets of active contacts that are suitable for each phase of each behavior are then discovered by the optimizer fully automatically. Additional innovations include a continuation scheme allowing helper forces at the potential contacts rather than the torso, as well as a feature-based model of physics which is particularly well-suited to the CIO framework.

2 Related Work

Early approaches to synthesizing human motion have been able to produce a wide repertoire of skills [Hodgins et al. 1995; Hodgins and Pollard 1997; Wooten and Hodgins 2000], but required expert manual specification for each new task. Since then, a lot of fruitful work has focused specifically on the task of locomotion. Simplified dynamical systems representing walking [Kajita et al. 2001; Kuo et al. 2005], running [Seipel and Holmes 2005], push recovery [Pratt et al. 2006] and uneven terrain navigation [Manchester et al. 2011] have been proposed that are well suited to analysis and control. The results were extended to full-detail bipeds [Yin et al. 2007; chi Wu and Popovic 2010; de Lasa et al. 2010] and quadrupeds [Coros et al. 2011]. [Srinivasan and Ruina 2005; Mordatch et al. 2010] have tried to capture different modes of locomotion within a single controller, though they still assume a fixed pattern of foot contacts that are specific to locomotion. Others proposed to intelligently combine individual controllers [Faloutsos et al. 2001; da Silva et al. 2009; Coros et al. 2009; Muico et al. 2011], although they still rely on existence of base controller libraries.

A more general approach to motion synthesis has been to pose the problem as trajectory optimization, subject to user constraints [Witkin and Kass 1988]. However, for all but the simplest problems the energy landscape of the optimization is very high-dimensional, prone to many local minima, and even discontinuous due to contact phenomena. To alleviate these issues, many approaches make use of human motion capture data to restrict the search space around stereotypical motions and pre-specify contact events [Popovic and Witkin 1999; Fang and Pollard 2003; Safonova et al. 2004; Liu et al. 2005]. [Liu et al. 2006] adapts motion capture data and contacts to multi-character interactions. By optimizing contact times for cyclic patterns, [Wampler and Popovic 2009] is able to synthesize gaits for a wide variety of non-humanoid characters from scratch.

Rather than placing restrictions on the optimization problem to solve it, several methods have focused on shaping the energy landscape to be smoother and better behaved. To widen the solution feasibility region, [Van De Panne and Lamouret 1995] initially use unphysical helper forces to aid the character and reduce them as optimization progresses, while [Yin et al. 2008] parametrizes the difficulty of the task itself. [Brubaker et al. 2009; Todorov 2011] reformulate contact as a smooth, rather than discontinuous phenomenon, where contact forces are always active, but smoothly diminish with the distance to the ground. The latter method has demonstrated success for continuous optimization of cyclic gaits [Erez et al. 2011]. [Jain and Liu 2011] accurately models characters with soft tissue, and shows improvements in stability and robustness for simple tracking algorithms. However, applying these formulations to general trajectory optimization problems remains difficult, because contact decisions are made implicitly as a (highly non-linear) function of the character's pose. Some methods (such as [Muico et al. 2009; Ye and Liu 2010; Liu 2009]) directly include contact forces as variables to be optimized, but they only have limited ability to manipulate contact state. By contrast, our contact variables make contact decisions explicit in the optimization.

Reasoning about contact events for navigation and object manipulation has also been considered by several planning approaches in robotics. [Kuffner et al. 2003; Chestnutt 2007; Hauser et al. 2008; Kolter et al. 2008] decompose the problem into two stages, first planning foot or hand placements and then synthesizing a motion trajectory that follows those placements. The two stages are only loosely coupled, and may not always produce the most efficient strategies. By contrast, our approach jointly plans both the contact events and the motion trajectory, and is able to exploit any synergies between the two.

The importance of temporally-extended actions and their potential to speed up optimization has been recognized in Reinforcement Learning, and has led to the Options framework [Sutton et al. 1999] which uses the formalism of Semi-Markov Decision Processes. In that framework however the temporally-extended actions (loosely corresponding to our movement phases) have to be specified in advance, while we discover them automatically.

3 Rationale and Overview

Our work was motivated by the observation that contact interactions are essential for most animal and human movements. Previous work on motion synthesis through numerical optimization has either pre-defined the contact interactions, or expressed them as functions of the movement trajectory and thereby optimized them indirectly, almost as a side effect of trajectory optimization. Our reasoning was that, if contacts are essential, they should play a more central role. This suggested optimizing over some auxiliary decision variables which directly describe when and where contacts are made. Our first attempts to develop such a method failed in interesting ways. We defined discrete variables specifying contacts between pairs of objects, and a continuous feedback control method that pushed the character along trajectories consistent with

this high-level contact specification. We found that, even though setting such discrete variables was relatively easy for humans (resulting in a novel way of motion scripting), optimizing them automatically was basically intractable. This was partly because discrete optimization is generally hard, and partly because it is difficult to tell what constitutes a good set of contacts without simultaneously considering the detailed trajectory that instantiates them. The moral of the story was that decisions regarding contact interactions should be encoded as continuous rather than discrete variables, and should be optimized simultaneously with the movement trajectory.

Continuous specification of desired contacts naturally leads to the idea of weights in cost functions. The contact-related auxiliary variables we use here (denoted $c_i \geq 0$ for contact i) have the following semantics: if c_i is large contact i must be active (i.e. the corresponding bodies must be touching), while if c_i is small we do not care what happens at contact i. Another important observation is that complex behaviors are naturally decomposed into phases, and the set of contacts remains invariant in each phase. The contact forces are not invariant (on the contrary, they change a lot) but the presence or absence of a contact is. This suggests making $c_{i,t}$ piece-wise constant over time, i.e. defining $c_{i,\phi(t)}$ where $\phi(t)$ is the movement phase at time t. The most direct approach at this point would be to define auxiliary cost terms weighted by $c_{i,\phi}$. If we did that, however, the optimizer will immediately set all c's to zero and effectively eliminate our auxiliary costs. One way to prevent this would be to constrain the sum of the c's, but this amounts to telling the optimizer how much overall contact it should use in a given behavior, and we do not know the answer in advance.

This leads to the next important realization – which is that our auxiliary variables c should also affect the dynamics, in such a way that setting them to zero would be suboptimal. Since they are associated with contacts, the natural way to enter the dynamics is to allow contact forces to be generated at contact i only when the corresponding c_i is large. This has another unexpected benefit: instead of finding the active contacts and performing various calculations that depend on the output of the collision detector (and are therefore non-smooth and difficult to optimize over), we can assume that the active contacts are those whose c's are large, resulting in simpler computations with smooth output.

The approach outlined above requires all potential contacts to be enumerated in advance, and an auxiliary variable c_i to be defined for each of them. If there are N bodies, we have N^2 potential contact pairs. To keep the problem size more manageable we introduce an unusual notion of contact which is one-sided, i.e. we associate the c's with individual bodies (called end-effectors below) rather than pairs of bodies. Thus we have N auxiliary variables instead of N^2. A potential drawback is that the law that "each action has equal and opposite reaction" no longer holds in a strict sense, although in practice the violations appear to be small and we have not found them to be problematic.

Finally, we introduce a simplified physics model consistent with our contact-centric philosophy. Instead of parametrizing the joint-space configuration of the character and using forward kinematics to compute end-effector positions and orientations, we parameterize the end-effectors and use inverse kinematics to define the joint-space configuration. In this way the optimizer can work directly with the end-effectors to which the auxiliary variables are associated. Kinematic constraints (i.e. fixed limb sizes and joint limits) are enforced as costs, and the dynamics are simplified by assuming that all mass is concentrated at the torso. We do not yet know if this physics simplification was necessary, and will find out in future work. We do know however that the method as presented below exceeded our most optimistic expectations, and synthesized remarkably complex movements within a couple of minutes of CPU time.

4 Contact-Invariant Optimization

We now describe the CIO method in detail. We begin with a general formulation that can be adapted to different types of physics models and tasks. We then describe our specific simplification of physics, followed by details of the behavioral tasks and the numerical optimization procedure.

4.1 General formulation and contact-invariant cost

Let $\mathbf{s}$ denote the real-valued solution vector that encodes the movement trajectory and auxiliary variables. The trajectory can be represented directly by listing the sequence of poses, or by function approximators such as splines (which is what we use here). All we require is that the character pose $\mathbf{q}_t(\mathbf{s})$ is a well-defined function of $\mathbf{s}$ at each (discrete) point in time $1 \leq t \leq T$. The auxiliary variables $c_{i,\phi(t)}(\mathbf{s}) \geq 0$ are also included in $\mathbf{s}$. The overall movement time T is partitioned into K intervals/phases, and $1 \leq \phi(t) \leq K$ is the index of the phase to which time step t belongs. In our current implementation the number of phases is predefined and their durations are equal, although in principle these parameters can also be optimized in an outer loop. $1 \leq i \leq N$ is an index over "end-effectors". Here end-effector does not refer to an entire rigid body (e.g. a hand or a foot), but to a specific surface patch on one of the rigid bodies. These patches/end-effectors are the only places were contact forces can be exerted, as explained below. The function $\mathbf{p}_i(\mathbf{q}) \in \mathbb{R}^3$ returns the center of patch i.

The CIO method computes the optimal solution $\mathbf{s}^*$ by minimizing a composite objective function $L(\mathbf{s})$ in the form

$$L(\mathbf{s}) = L_{\mathrm{CI}}(\mathbf{s}) + L_{\mathrm{Physics}}(\mathbf{s}) + L_{\mathrm{Task}}(\mathbf{s}) + L_{\mathrm{Hint}}(\mathbf{s}) \quad (1)$$

L_{CI} is a novel contact-invariant cost introduced here. L_{Physics} penalizes physics violations; we enforce physical consistency using a soft cost rather than a hard constraint because this enables powerful continuation methods. L_{Task} specifies the task objectives, and is the only term that needs to be modified in order to synthesize a novel behavior. L_{Hint} is optional and can be used to provide hints (e.g. ZMP-like costs use here) in the early phases of optimization. Continuation methods are implemented by weighting these costs differently in different phases of optimization; see below.

The contact-invariant cost L_{CI} is defined as

$$L_{\mathrm{CI}}(\mathbf{s}) = \sum_t c_{i,\phi(t)}(\mathbf{s}) \left(\|\mathbf{e}_{i,t}(\mathbf{s})\|^2 + \|\dot{\mathbf{e}}_{i,t}(\mathbf{s})\|^2 \right) \quad (2)$$

$\mathbf{e}_{i,t}$ is a 4D contact-violation vector for end-effector i at time t. Recall that a large value of $c_{i,\phi(t)}$ means that end-effector i should be in contact with the environment during the entire movement phase $\phi(t)$ to which time t belongs. Thus when c is large we want the corresponding $\mathbf{e}$ to be small. This vector encodes misalignment in both position and orientation. The first 3 components of $\mathbf{e}$ are the difference vector between the end-effector position $\mathbf{p}_i(\mathbf{q}_t)$ and the "nearest point" on any surface in the environment (including other body segments). The last component of $\mathbf{e}$ is the angle between the surface normal at the nearest point and the surface normal at the end-effector. The cost L_{CI} penalizes both $\mathbf{e}$ and its velocity $\dot{\mathbf{e}}$ which corresponds to slip.

Our definition of "nearest point" is unusual in an important way: we effectively use a soft-min instead of a min operator. Let $\mathbf{n}_j(\mathbf{p})$ denote the actual nearest point to $\mathbf{p}$ on surface j. Define the weights

$$\eta_j(\mathbf{p}) = \frac{1}{1 + \|\mathbf{p} - \mathbf{n}_j(\mathbf{p})\|^2 k} \quad (3)$$

where $k = 10^4$ is a smoothness parameter. The nearest point is then obtained by normalizing the weights η_j to sum to 1, and computing the weighted average of the $\mathbf{n}_j$'s. Intuitively, when $\mathbf{p}_i$ is far from any surface (and c_i is large), the cost L_{CI} will push it towards some average of the surface "mass". When $\mathbf{p}_i$ gets close to a surface, it will be pushed towards the nearest point on that surface. This construction also makes L_{CI} smooth, which facilitates numerical optimization.

4.2 Inverse dynamics and physics-violation cost

Next we describe the physics-violation cost L_{Physics}. This cost has two components. One is general and depends on our novel formulation of contact dynamics which is essential to the CIO method. The other is specific to the simplified model of multi-body dynamics described below – which can be replaced with a full physics model without modifying the rest of the method. In this subsection we focus on a single time step and omit the time index t.

Let $\mathbf{f} \in \mathbb{R}^{6N}$ denote the vector of contact forces acting on all N end-effectors. We have a 6D vector per contact because we allow torsion around the surface normal, and furthermore the origin of the contact force is allowed to move inside the end-effector surface patch. Thus, unlike previous work which models contact as a sum a multiple contact points, we model an entire contact surface patch, allowing us to reason about contact at a coarser scale and speed up the optimization. Note that all potential contacts are considered at all times, and so the dimensionality of the contact force vector remains constant, resulting in smoothness of the cost. Contact forces here are associated with individual end-effectors and not with pairs of contacting bodies. We only model contacts at pre-defined end-effectors, although our definition is sufficiently general to include surface patches on any body part.

Let $J(\mathbf{q}) \in \mathbb{R}^{6N \times D}$ denote the Jacobian matrix mapping generalized velocities $\dot{\mathbf{q}}$ to contact-space velocities for the contact model described above; $D = \dim(\dot{\mathbf{q}})$ is the number of degrees of freedom in the character. Let $\tau(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ denote the inverse of the smooth dynamics, which equals the sum of contact and applied forces:

$$\tau(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = J(\mathbf{q})^T \mathbf{f} + B\mathbf{u} \tag{4}$$

Here $\mathbf{u}$ is the vector of applied forces/controls in the actuated space. They are mapped to the full space by the matrix B, which has zeros in the rows corresponding to "root" forces/torques acting on the torso or on passive objects. The smooth inverse dynamics are

$$\tau(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \tag{5}$$

where M is the inertia matrix, C the matrix of Coriolis and centrifugal terms, and $\mathbf{g}$ is gravity.

Our goal now is to complete the inverse dynamics computation, and in particular recover $\mathbf{f}, \mathbf{u}$ given τ, J, B. We do this by minimizing the squared residual in (4) subject to friction-cone constraints on $\mathbf{f}$ and quadratic regularization for $\mathbf{f}$ and $\mathbf{u}$. Thus we have the following quadratic programming (QP) problem:

$$\begin{aligned} \mathbf{f}, \mathbf{u} = \quad & \arg\min_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{u}}} \left\| J^T \widetilde{\mathbf{f}} + B\widetilde{\mathbf{u}} - \tau \right\|^2 + \widetilde{\mathbf{f}}^T W \widetilde{\mathbf{f}} + \widetilde{\mathbf{u}}^T R \widetilde{\mathbf{u}} \\ & \text{subject to} \quad A\widetilde{\mathbf{f}} \leq \mathbf{b} \end{aligned} \tag{6}$$

The pose-dependent matrix $A(\mathbf{q})$ and vector $\mathbf{b}(\mathbf{q})$ encode the standard pyramid approximation to the friction cone which makes the inequality constraints linear. They also include linear inequality constraints that restrict the origin point of each contact force to the surface patch of its corresponding end-effector. Currently these patches are rectangles. For end-effectors that are allowed to grab (i.e. hands) we omit the friction-cone constraints, allowing both positive and negative normal forces.

The control regularization matrix R is constant and s.p.d. The contact force regularization matrix W depends on the auxiliary variables: W is diagonal and its 6 diagonal elements corresponding to the i-th contact force vector are

$$W_{j,j} = \frac{k_0}{c_i^2 + k_1} \tag{7}$$

for all j between $6i - 5$ and $6i$. We used $k_0 = 10^{-2}, k_1 = 10^{-3}$. The components of W corresponding to hand end-effectors were four times larger, because hands are generally weaker than feet.

The quadratic program (6) is convex and therefore has a unique solution – which can be found using a number of algorithms. Here we used a specialized sequential solver due to [Lee and Goswami 2010] because it is easy to implement efficiently, although the results obtained with an off-the-shelf QP solver were very similar.

In summary, inverse dynamics are computed by first computing the quantities $\tau(\mathbf{s}), J(\mathbf{s}), A(\mathbf{s}), \mathbf{b}(\mathbf{s}), W(\mathbf{s})$ as described above, and then solving (6) which yields $\mathbf{f}(\mathbf{s})$ and $\mathbf{u}(\mathbf{s})$. Re-introducing the time index t, the physics-violation cost is

$$L_{\text{Physics}}(\mathbf{s}) = \sum_t \left\| J_t(\mathbf{s})^T \mathbf{f}_t(\mathbf{s}) + B\mathbf{u}_t(\mathbf{s}) - \tau_t(\mathbf{s}) \right\|^2 \tag{8}$$

Note that even if this cost can be reduced to zero by a certain choice of $\mathbf{f}$ and $\mathbf{u}$, the actual $\mathbf{f}$ and $\mathbf{u}$ computed in (6) will generally be different because of the extra regularization terms.

4.2.1 Trade-off between L_{CI} and L_{Physics}

Let us now examine how the two cost terms L_{CI} and L_{Physics} defined in (2) and (8) are affected by the auxiliary variables c. The term L_{CI} achieves its global minimum of zero when all c's are set to zero. However this makes the W matrix in (7) large, contact forces become expensive, and the QP solver for (6) prefers to violate physics rather than use large contact forces – which in turn leads to a substantial L_{Physics} term. Conversely if all c's are large, the QP solver is able to reconcile any trajectory with the inverse dynamics model by generating contact forces at all end-effectors, including those that are not actually in contact. This makes L_{Physics} small, but now L_{CI} is large because end-effectors that are not in contact have large c's. Thus the optimal trade-off is achieved when c_i is large only for end-effectors that are in contact, and furthermore the trajectory can be generated using contact forces only at those end-effectors. In other words, at the optimal solution the auxiliary variables c correspond to the contacts that end up being active.

Since the controls $\mathbf{u}$ are constrained to act in the actuated space, root helper forces (often used in prior work for continuation) are not allowed here. If we were to allow such forces the above trade-off in terms of c would no longer hold. On the other hand, the ability of our method to generate contact forces from a distance provides an even more effective continuation method: wishful thinking in the early phases of optimization is still possible, but it is always associated with potential contacts, making it easier to transition to physically-realistic contact dynamics later in optimization.

4.2.2 Simplified physics model

While the CIO method as described above can be used with standard physics models as implemented in existing physics engines, our implementation relies on a simplified model yielding a favorable trade-off between physical realism and optimization efficiency.

Instead of representing the pose $\mathbf{q}$ directly and then computing the end-effector positions $\mathbf{p}_i(\mathbf{q})$ using forward kinematics, we represent the end-effector positions as well as their orientations directly (i.e. as functions of spline parameters contained in $\mathbf{s}$) and then define the pose $\mathbf{q}$ using inverse kinematics; see Appendix. All mass is assumed to be concentrated at the root bodies: the torso of each character, as well as any passive objects. Non-smooth movements of the (now-massles) limbs are avoided by including an acceleration cost described later. The inverse dynamics are still in the form (4) and the quadratic program defining the contact force and control is still in the form (6), but all computations are now simplified.

Representing the pose in terms of end-effector positions and orientations makes it difficult to enforce kinematic constraints exactly. However we turn this to our advantage, by introducing an additional continuation method that allows limbs to stretch and joint limits to be violated early in optimization. This is done by adding quadratic costs to L_{Physics}, that penalize any deviations of the limb lengths from their reference values as well as any joint limit violations. We also penalize penetration of the character's bodyparts (approximated for collision with capsules) against the environment, or other bodyparts.

4.3 High-level goals and task cost

The cost $L_{\text{Task}}(\mathbf{s})$ encodes the high-level goals of the movement. It includes task-specific terms specifying the desired outcome, and generic terms (integrated over time) specifying that the movement should be energy-efficient and smooth:

$$L_{\text{Task}}(\mathbf{s}) = \sum_b \ell_b(\mathbf{q}_T(\mathbf{s})) + \sum_t \|\mathbf{f}_t(\mathbf{s})\|^2 + \|\mathbf{u}_t(\mathbf{s})\|^2 + \|\ddot{\mathbf{q}}_t(\mathbf{s})\|^2 \tag{9}$$

Here ℓ_b are task-specific terms which only depend on the final pose $\mathbf{q}_T$, and b is an index over different tasks. Several tasks can be composed together, such as combining a standing task with the moving to target task. We use the above general form of L_{Task} for all tasks except for kicking/punching. In that case we specify an ℓ_b at regular intervals when each target should be hit, and also include dependence on $\dot{\mathbf{q}}$ because we want the targets to be hit with a certain end-effector velocity.

The general procedure for constructing the task-specific costs ℓ_b is to identify a vector of positional (and optionally velocity) features $\mathbf{h}_b(\mathbf{q})$ that are key to task b, define the desired feature values $\mathbf{h}_b^*$ at the end of the movement (or at other important points in time such as target hits), and then construct ℓ_b as

$$\ell_b(\mathbf{q}_T(\mathbf{s})) = \|\mathbf{h}_b(\mathbf{q}_T(\mathbf{s})) - \mathbf{h}_b^*\|^2 \tag{10}$$

In this way, a final position task ℓ_{pos} can be specified by using $\mathbf{h}_{\text{pos}}$ that selects torso position, and setting $\mathbf{h}_{\text{pos}}^*$ to the desired position. Final orientation task ℓ_{dir} can be defined similarly for torso facing direction. Standing task ℓ_{stand} can be expressed by using a combination of $\mathbf{h}_{\text{stand}}$ and $\mathbf{h}_{\text{stand}}^*$ which specifies that the center of torso should be between two feet, the feet be fully extended, and the torso direction be aligned with the vertical direction vector.

The relative importance of the different features can be adjusted by scaling the corresponding elements of $\mathbf{h}$.

4.4 Heuristic sub-goals and hint cost

In the absence of good initialization – which in the present context would correspond to motion capture data or other detailed user inputs we aim to avoid – numerical optimization can be sped up by providing heuristic sub-goals early on, and then disabling them near convergence. Such heuristics (also known as shaping) are not meant to be part of the true cost, but rather guide the solution to a region from where the true cost can be optimized efficiently. We found that even though most of the behaviors we studied could be synthesized without such heuristics, in some cases (particularly those involving two characters) a certain type of heuristic helps. This heuristic is based on the ZMP stability criterion used in locomotion, where the objective is to keep the "zero moment point" $\mathbf{z}(\mathbf{q}, \ddot{\mathbf{q}})$ in the convex hull of the support region [Vukobratovic and Borovac 2004]. Let $\mathbf{n}(\mathbf{z})$ denote the nearest (in a soft-min sense) to $\mathbf{z}$ point in the convex hull. We compute $\mathbf{n}$ by expressing it as a convex combination of the end-effector positions: $\mathbf{n} = \sum_i \lambda_i \mathbf{p}_i$ where $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, and solving for the coefficients λ using quadratic programming regularized by the same weights W as in (7). Then the hint cost is

$$L_{\text{Hint}}(\mathbf{s}) = \sum_t \max(\|\mathbf{z}_t(\mathbf{s}) - \mathbf{n}(\mathbf{z}_t(\mathbf{s}))\| - \epsilon, 0)^2 \tag{11}$$

This is a half-quadratic starting ϵ away from the convex hull. The parameter ϵ is used to adjust how strictly we want to enforce the ZMP stability criterion.

4.5 Numerical optimization and continuation

We optimize the composite cost $L(\mathbf{s})$ defined in (1) using an off-the-self implementation of the LBFGS algorithm. The dimensionality of the vector $\mathbf{s}$ is $(12(N+1)+N)K$, where again N is the number of end-effectors and K is the number of movement phases. For the description of the parameters given our simplified model, see Appendix A. We use K between 10 and 20 depending on the complexity of the task. Each phase lasts 0.5 sec. The inverse dynamics and cost are evaluated at 0.1 sec intervals (note that the analytical spline representation allows us to evaluate the dynamics and cost at any point in time). The gradient $\nabla L(\mathbf{s})$ which is needed for numerical optimization is approximated using finite differences (with $\epsilon = 10^{-3}$). Our implementation of finite differences takes advantage of the fact that many of the cost terms depend only on the pose at a single point in time, and do not need to be recomputed when the rest of the trajectory is perturbed.

Continuation is implemented by weighting the four terms in (1) differently in different phases of the optimization process (not to be confused with movement phases). The optimization process has three phases as follows. In Phase 1 only L_{Task} is enabled. This causes the optimizer to rapidly discover a movement that achieves the task goals without being physically realistic. In Phase 2 we enable all four terms, except L_{Physics} is down-weighted by 0.1 so that physical consistency is enforced gradually. In Phase 3 we fully enable all terms except for L_{Hint} – which is no longer needed and is undesirable at this point, because we do not want it to affect the final solution. Qualitatively, Phase 1 corresponds to rapid discovery combined with wishful thinking; Phase 2 corresponds to cautious enforcement of physical realism while being guided by optional hints; Phase 3 corresponds to refinement of the final solution. The solution obtained at the end of each phase is perturbed with small zero-mean Gaussian noise (to break any symmetries) and used to initialize the next phase. The initialization for Phase 1 is completely uninformative – a static initial pose. We found that using such continuation is often important. The good news is that exactly the same continuation scheme was successful in all of the diverse behaviors we studied, and so our method does not need behavior-specific adjustments.

Each stage of the optimization takes between 250-1000 iterations, depending on the complexity of the problem. The total time to synthesize each animation clip ranges between 2 to 10 minutes on a quad-core 2.3GHz Intel Xeon machine.

5 Results

By using the tasks ℓ_b described in section 4.3 we can synthesize a wide variety of behaviors from only a small set of high-level goals.

Getting Up, Walking, Climbing By using only ℓ_{stand} described in section 4.3 and using an initial pose of the character lying on the ground, we synthesize the motion of getting up. Different initial poses (such as lying on the back, or on the stomach) result in different getting up strategies. If the character is initially standing, ℓ_{stand} is satisfied by simply continuing to stand. By using ℓ_{stand} task in combination with ℓ_{pos} task and an initially-standing pose, we induce walking. The pattern of footsteps typical to walking is not specified and emerges automatically from our optimization. ℓ_{pos} task is shown in the corresponding video with a white crosshair.

By using only ℓ_{stand} and ℓ_{pos} tasks, but changing the environment to include an obstacle, a range of strategies emerges from simply stepping over the obstacle, to using hands to prop the character up, to using hands to grip and climb a really tall obstacle. The coefficient of friction μ on the feet is 2 to allow foot plants on completely vertical slopes.

Handstands, Punches, Kicks Modifying ℓ_{stand} by swapping feet for hands in the specification and commanding that feet should point upwards, we create a handstand task $\ell_{\text{handstand}}$. When only this task is active and the character is initially standing, they will make a series of preparatory movements and prop themselves up onto their hands. $\ell_{\text{handstand}}$ and similarly be combined with ℓ_{pos} and ℓ_{rot} tasks.

By using a task that specifies positions and velocities for limb end effectors at various points in the trajectory, we generate striking motions such as punches and kicks. Using only this task and no others, the character knows the strike targets in advance and appropriately chooses their movement strategy.

Non-Human Character Morphologies Our optimization is not specific to humans and generalizes to different morphologies. We consider a human with a such as very wide humanoids, or quadrupeds for the tasks above. Natural trot pattern of footsteps emerges for locomotion tasks with quadrupeds.

Interaction with Objects Additional objects can be introduced into the world and their trajectories included as variables in the optimization. An object has auxiliary contact variables for every character's end effector, indicating that the end effector is gripping the object. The object is then able to generate contact forces and move around, but L_{CI} cost applies except that now end effector has to be touching the surface of the object.

From only a single goal on the terminal location of the object, the strategy of the character having to pick up and carry the object to the destination emerges. Again, no other tasks are specified.

Interaction Between Characters Our algorithm can be extended to jointly optimizing for the motion of multiple characters, resulting in cooperation between them. For the task of moving the object above, multiple characters distribute the workload and pass the object from one to the other. We can control the workload distribution by controlling the penalty on contact forces in equation (9) for one of the characters, and making the other character do most of the object carrying work.

By only setting a single goal on the target height of one of the characters, the two characters cooperate and one climbs on top of the other.

6 Conclusion and Future Work

In this paper we presented a fully automated framework for synthesizing a wide range of movement behaviors, and demonstrated its effectiveness on complex and rarely studied behaviors. Our framework is agnostic to the morphology of the character, and indeed we showed that movement behaviors can be created for significantly different types of characters. The contact-invariant optimization method could likely be applied to other domains where constraint-driven phases bifurcate the optimization space making it very hard to find solutions numerically. We developed an effective continuation scheme suitable for our optimization method.

We also introduced a feature-based physics model that allows for efficient consideration of dynamic aspects in the inner loop of the optimization procedure. This relaxation naturally comes at a cost. We model the character's kinematics and contact interactions with the environment in detail, but represent its dynamics as a single rigid body and do not model the limb dynamics. This simplification effectively results in limbs with infinitely strong muscles (and no penalty for using them), which in turn can lead to energy-inefficient motions such as using bent knees for support, or having arms extended against gravity. This limitation can be removed by using full-body inverse dynamics to calculate the character's joint torques, and penalizing the torques or some related quantity.

The style of the motions was not the focus of this work, and could use a lot of improvement, particularly for low-energy motions such as walking where humans use every bit of physiology (which we do not model) to their advantage. Since our method performs long-horizon trajectory optimization, we can easily incorporate biomechanically-inspired objective terms from [Wang et al. 2009] to shape the stylistic aspects of the motion.

In all our examples, standard methods for local gradient-based optimization were able to find good solutions efficiently. This is one of the key advantages of our framework. Still, the use of global optimization could provide more robust exploration of the space of motions, especially for tasks such as getting up that have a wide range of possible and equally good solutions. In such cases we would prefer to produce multiple solutions, and select the ideal one.

Another key advantage of our framework is the simultaneous optimization of contacts and smooth portions of the movement. This was made possible by introducing an auxiliary decision variable for each potential contact, and keeping these variables constant within each phase. As a result, we were able to optimize very long and temporally complex movement sequences that have previously remained beyond the reach of numerical optimization methods. The price we had to pay was that the potential contact points (or rather patches) had to be pre-defined. We are of course penalizing penetrations everywhere, but this is not the same as actively optimizing over active contacts. One way to remove this limitation is to simply increase the number of potential contacts and cover the entire body with sufficient density. It remains to be seen how this will affect CPU time. Since we are using finite-differencing to compute derivatives, the number of cost evaluations will grow linearly with the number of auxiliary variables. The size of the quadratic program that needs to be solved in the inner loop will also increase, but since it corresponds to a convex optimization problem, this is not a cause for concern. The key question is how the number of iterations of the main optimization loop will be affected. Introducing additional variables can make an optimization problem easier or harder depending on the nature of the problem, and so this question will have to be answered empirically.

In the examples presented here the number of phases was fixed. We found that the character tends to balance the task across all phases that are given to it. As long as the number of phases was sufficient, their exact number had little effect on the solution. The number and duration of phases could also be optimized.

Perhaps the most exciting direction for future work is applying CIO to a full physics model that takes into account the limb inertias and non-linear interaction forces. Indeed we formulated the CIO method so that it is directly applicable to such a model. We expect this to significantly enhance the realism/style of the resulting movements, particularly in behaviors where saving energy is important. While we do not anticipate any significant obstacles, how efficient the method will be in the context of these more challenging optimization problems remains to be seen. It may turn out that a hybrid approach is preferable, where we first use the present simplified model to obtain a solution that already looks quite good, and then optimize with respect to a full physics model to refine the solution. Alternatively it may turn out that CIO is just as efficient (modulo the increased CPU time needed to evaluate the more complex inverse dynamics), in which case it could be applied directly.

Acknowledgments

References

BRUBAKER, M. A., SIGAL, L., AND FLEET, D. J. 2009. Estimating contact dynamics. In *ICCV*, 2389–2396.

CHESTNUTT, J. 2007. *Navigation Planning for Legged Robots*. PhD thesis, Carnegie Mellon University.

CHI WU, J., AND POPOVIC, Z. 2010. Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph. 29*, 4.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust task-based control policies for physics-based characters. *ACM Trans. Graph. 28*, 5.

COROS, S., KARPATHY, A., JONES, B., REVÉRET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. *ACM Trans. Graph. 30*, 4, 59.

DA SILVA, M., DURAND, F., AND POPOVIC, J. 2009. Linear bellman combination for control of character animation. *ACM Trans. Graph. 28*, 3.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-Based Locomotion Controllers. *ACM Trans. Graphics 29*, 3.

EREZ, T., TASSA, Y., AND TODOROV, E. 2011. Infinite-horizon model predictive control for periodic tasks with contacts. In *Robotics: Science and Systems*.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *SIGGRAPH*, 251–260.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. Graph. 22*, 3, 417–426.

GRASSIA, F. S. 1998. Practical parameterization of rotations using the exponential map. *J. Graph. Tools 3* (March), 29–48.

HAUSER, K. K., BRETL, T., LATOMBE, J.-C., HARADA, K., AND WILCOX, B. 2008. Motion planning for legged robots on varied terrain. *I. J. Robotic Res. 27*, 11-12, 1325–1349.

HODGINS, J. K., AND POLLARD, N. S. 1997. Adapting simulated behaviors for new characters. In *SIGGRAPH*, 153–162.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *SIGGRAPH*, 71–78.

JAIN, S., AND LIU, C. K. 2011. Controlling physics-based characters using soft contacts. *ACM Trans. Graph. (SIGGRAPH Asia) 30* (Dec.), 163:1–163:10.

KAJITA, S., MATSUMOTO, O., AND SAIGO, M. 2001. Real-time 3D walking pattern generation for a biped robot with telescopic legs. In *Proc. ICRA*, 2299–2306.

KOLTER, J. Z., RODGERS, M. P., AND NG, A. Y. 2008. A control architecture for quadruped locomotion over rough terrain. In *ICRA*, 811–818.

KUFFNER, J. J., NISHIWAKI, K., KAGAMI, S., INABA, M., AND INOUE, H. 2003. Motion planning for humanoid robots. In *ISRR*, 365–374.

KUO, A. D., DONELAN, J. M., AND RUINA, A. 2005. Energetic consequences of walking like an inverted pendulum: step-to-step transitions. *Exercise and sport sciences reviews 33*, 2 (Apr.), 88–97.

LEE, S.-H., AND GOSWAMI, A. 2010. Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In *IROS*, 3157–3162.

LIU, C. K., HERTZMANN, A., AND POPOVIC, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph. 24*, 3, 1071–1081.

LIU, C. K., HERTZMANN, A., AND POPOVIC, Z. 2006. Composition of complex optimal multi-character motions. In *Symposium on Computer Animation*, 215–222.

LIU, C. K. 2009. Dextrous manipulation from a grasping pose. *ACM Trans. Graph. 28*, 3.

MANCHESTER, I. R., METTIN, U., IIDA, F., AND TEDRAKE, R. 2011. Stable dynamic walking over uneven terrain. *I. J. Robotic Res. 30*, 3, 265–279.

MORDATCH, I., DE LASA, M., AND HERTZMANN, A. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Trans. Graph. 29*, 4.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware Nonlinear Control of Dynamic Characters. *ACM Trans. Graphics 28*, 3, 81.

MUICO, U., POPOVIC, J., AND POPOVIC, Z. 2011. Composite control of physically simulated characters. *ACM Trans. Graph. 30*, 3, 16.

POPOVIC, Z., AND WITKIN, A. P. 1999. Physically based motion transformation. In *SIGGRAPH*, 11–20.

PRATT, J., CARFF, J., AND DRAKUNOV, S. 2006. Capture point: A step toward humanoid push recovery. In *in 6th IEEE-RAS International Conference on Humanoid Robots*, 200–207.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph. 23*, 3, 514–521.

SEIPEL, J. E., AND HOLMES, P. 2005. Running in three dimensions: Analysis of a point-mass sprung-leg model. *I. J. Robotic Res. 24*, 8, 657–674.

SRINIVASAN, M., AND RUINA, A. 2005. Computer optimization of a minimal biped model discovers walking and running. *Nature* (Sept.).

STEPHENS, B. 2011. *Push Recovery Control for Force-Controlled Humanoid Robots*. PhD thesis, Carnegie Mellon University.

SUTTON, R., PRECUP, D., AND SINGH, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence 112*, 181–211.

TODOROV, E. 2011. A convex, smooth and invertible contact model for trajectory optimization. In *ICRA*, 1071–1076.

VAN DE PANNE, M., AND LAMOURET, A. 1995. Guided optimization for balanced locomotion. In *6th Eurographics Workshop on Animation and Simulation, Computer Animation and Simulation, September, 1995*, Springer, Maastricht, Pays-Bas, D. Terzopoulos and D. Thalmann, Eds., Eurographics, 165–177.

VUKOBRATOVIC, M., AND BOROVAC, B. 2004. Zero-moment point - thirty five years of its life. *I. J. Humanoid Robotics 1*, 1, 157–173.

WAMPLER, K., AND POPOVIC, Z. 2009. Optimal gait and form for animal locomotion. *ACM Trans. Graph. 28*, 3.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing Walking Controllers. *ACM Trans. Graphics 28*, 5, 168.

WITKIN, A., AND KASS, M. 1988. Spacetime Constraints. In *Proc. SIGGRAPH*, vol. 22, 159–168.

WOOTEN, W. L., AND HODGINS, J. K. 2000. Simulating leaping, tumbling, landing, and balancing humans. In *ICRA*, 656–662.

YE, Y., AND LIU, C. K. 2010. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph. 29*, 4.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. *ACM Trans. Graph. 26*, 3, 105.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Trans. Graph. 27*, 3.

A Simplified Character Model

Our simple model specifies character's state $\mathbf{q}(\mathbf{s})$ at a particular time through a small number of features, rather than a full set of joint angles.

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_c & \mathbf{r}_c & \mathbf{p}_{1\ldots N} & \mathbf{r}_{1\ldots N} \end{bmatrix}^T \tag{12}$$

Where $\mathbf{p}_c$ and $\mathbf{r}_c$ are torso position and orientation, respectively, and $\mathbf{p}_i$, $\mathbf{r}_i$ are end effector positions and orientations for each limb i (see figure 2). Rotations are represented with exponential map [Grassia 1998] because of its suitability in trajectory optimization.

From the above features, we can reconstruct the actual character's pose, including limb base locations $\mathbf{b}_i$, which can be derived from local location points on the torso. We assume character's limbs have two links, which allows us to analytically solve for middle joint location $\mathbf{m}_i$ and orientations of the two links. For limbs that have more than two links, it would be necessary to use an iterative inverse kinematics method to derive the individual joint locations.

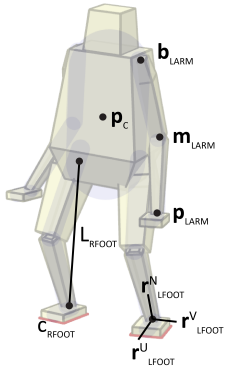

Figure 2: Simplified Character Model. *The features used in our character description with collision capsule geometry overlaid.*

We define the motion with positions and velocities of our features at the boundaries between phases. Cubic splines are used to define a continuous feature trajectory from which positions, velocities, accelerations at any point in the trajectory can be computed. Combining contact variables for the phase into a vector $\mathbf{c}$, the solution vector $\mathbf{s} \in \mathbb{R}^{(12(N+1)+N)K}$ that we optimize is

$$\mathbf{s} = \begin{bmatrix} \mathbf{x}_{1\ldots K} & \dot{\mathbf{x}}_{1\ldots K} & \mathbf{c}_{1\ldots K} \end{bmatrix}^T \tag{13}$$

The dynamics of our simple model correspond to those of a single rigid body with multiple forces acting on it from rectangular contact surfaces. In this setting, contact forces can efficiently be solved using either the approach of [Stephens 2011] or [Lee and Goswami 2010].