Force-Based Motion Editing for Locomotion Tasks

Nancy S. Pollard
Brown University

Fareed Behmaram-Mosavat
Brown University

Abstract

This paper describes a fast technique for modifying motion sequences for complex articulated mechanisms in a way that preserves physical properties of the motion. This technique is relevant to the problem of teaching motion tasks by demonstration, because it allows a single example to be adapted to a range of situations. Motion may be obtained from any source; for example, it may be captured from a human user. A model of applied forces is extracted from the motion data, and forces are scaled to achieve new goals. Each scaled force model is checked to ensure that frictional and kinematic constraints are maintained for a rigid body approximation of the character. Scale factors can be obtained in closed form, and constraints can be approximated analytically, making motion editing extremely fast. To demonstrate the effectiveness of this approach, we show that a variety of simulated jumps can be created by modifying a single keyframed jumping motion. We also scale a simulated running motion to a new character and to a range of new velocities.

1 Introduction

Programming a motion task by demonstration is an appealing goal, because it limits the expertise required of the user. User provided examples are "expensive," however, and will only sparsely cover the task space. For these reasons, much previous work has been focused on extracting task objectives from example motion for use in control, learning, or planning processes that can fill the gaps between examples. Task objectives may include impedance [1], motion of a manipulated object [2], or a symbolic representation of the task [9]. (See [4] for an overview.)

Most previous research in programming by demonstration has been directed towards manipulating or applying force to an external object. This paper applies this idea to simulation of locomotion tasks (e.g. running and jumping). Here, the role of forces is not to control an external object, but to control the motion of the character itself. Task objectives are positions and velocities of the center of mass at key points in the motion (Figure 2). Ground contact forces are manipulated to alter those objectives, for example to create runs at different speeds or jumps of different heights. Friction and kinematic constraints are incorporated so that forces remain within a given friction cone and the contact point is always reachable. We focus on motion transformations that can be done in closed form – i.e. those that do not require a search over parameter space.

Contributions of the paper are:

- A task model for locomotion tasks that can be extracted from any source of example motion.
- Fast techniques for editing this task model to achieve new goals.
- A closed form approximation to basic frictional and kinematic constraints to support fast constraint checking.

These techniques are combined in an interactive motion editor, but could also be used in a motion planning algorithm. The research was performed with the short term goal of animating human characters in a physically plausible way, but it could also be a first step toward developing flexible behaviors for a humanlike robot.

2 Background

A variety of techniques have been used to create motion for complex characters or robots, including interpolation of examples, constrained optimization, and development of task-specific control systems. Interpolation or blending can be used when many motion examples are available. In animation, interpolation is often performed on joint angle curves (e.g.[18]). These techniques are kinematic, and so the resulting motion may not be physically plausible. Some alternatives are to use physically-based simulation to train neural networks (e.g. [5]) or to interpolate control parameters. The latter approach has been used in conjunction with memory-based or nonparametric techniques for learning from experimental data (e.g. [2]). Generating physically correct results using interpolation is challenging, however, when the state space dimension is high.

If an expression for optimal motion is available, constrained optimization techniques can be used, as in

Figure 1: This paper describes a fast technique for editing motion. This technique has been used to change running velocity, scale running motion to new characters, and change the height and distance of a jump. The running motion was obtained from a physically-based simulation [8]. The jumping motion was keyframed.

[20][13][14]. Optimization can generate physically plausible results when the character model is physically-based (e.g. when a simulation is used inside the optimization loop). Its main disadvantage is computation time, although simplifying approaches [14] can help to alleviate this problem.

If a control system is available, some flexibility will be designed into that control system [16] [15] [6] [19] [11] [8]. Creation of robust and flexible control systems for bipedal locomotion is difficult, however, and requires the skills of an experienced designer.

Rigid body approximations have been used for analysis of both hopping robots [16] [10] and running humans [12]. This paper builds on this work, using a rigid body approximation to ensure that basic frictional and kinematic constraints are met for any new motion that is generated. Kinematic interpolation is used to create the motion of the rest of the body. This two stage process gives us some of the physical plausibility generally associated with optimization or control, along with the ability to interactively create new motion that is a feature of kinematic techniques (Figure 1).

3 Stances

For this paper, we segment motion into stances (Figure 2). For dynamic motions such as running and jumping, each stance is a period of continuous ground contact, and

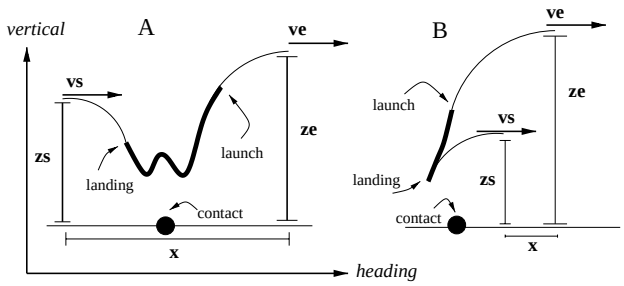

Figure 2: (A) Example motion is segmented into stances. A stance is defined based on the motion of a character's center of mass. Parameters are start and end heights zs and ze, start and end velocities vs and ve in the *heading* direction, end velocity vp in the *perpendicular* direction (into the page, not shown) and distance x from the apex of the first flight phase to the apex of the second flight phase. (B) The apex used to define a stance might not be reached in the actual motion.

stances are separated by periods of flight.

A stance is defined with six parameters (Figure 2A). The curve in the figure represents the center of mass of the character. A stance has its own coordinate frame, with axes in the *heading*, *perpendicular*, and *vertical* directions. The *heading* of a stance is the direction of the character's horizontal velocity at the time of landing. Parameters describe position and velocity at the apices of the ballistic portions of a stance. The parameters are: start velocity (vs), start height (zs), end velocity in the *heading* direction (ve), end velocity in the *perpendicular* direction (vp), end height (ze), and distance from apex to apex in the *heading* direction (x). Parameters zs and ze are defined with respect to the height of the contact point at landing. This particular set of parameters was chosen because they can be controlled easily by scaling applied forces (Section 4), and because they are visually intuitive. Note that the ballistic portion of the stance as sketched in Figure 2A may not be present in the actual motion. In walking, for example, the character is in constant contact with the ground. The theoretical apex is always used to define the stance, however (Figure 2B).

Stance parameters can be extracted from motion data and a physical description of the creature. The results in this paper were all generated from body root position, body root orientation, and joint angles provided at 30 frames per second, a geometric description of the creature, and density information [3].

Orientation is omitted from the set of parameters used to define a stance under the assumption that orientation can, to some extent, be controlled independently of position. In the examples in this paper, we partially control orientation by searching for an "optimal" contact point. Sections 6 and 7 describe some of the problems we encountered with

controlling orientation and outline a possible solution.

3.1 Forces

A model of applied forces is extracted for each stance. Applied forces are modeled with the dominant terms of the discrete Fourier transform (DFT) of sampled force data. The center of mass position for an entire motion is numerically differentiated twice and fit with a spline to obtain acceleration $\underline{\ddot{x}}(t)$. Start and end times for a stance are identified, and the *heading* direction for that stance is computed from the motion data. Acceleration is rotated into the *vertical / heading / perpendicular* coordinate system, multiplied by character mass m to obtain total force, and sampled at N discrete timesteps:

$$\underline{F}_T(j) = m\underline{\ddot{x}}\left(t_0 + \frac{j\ \Delta t}{N}\right), \quad j = 0, ..., N-1 \quad (1)$$

where the stance phase begins at time t_0 and ends at time $(t_0 + \Delta t)$. An approximate continuous function for total force $\underline{f}_T(\tau)$ is then obtained from the dominant terms of the discrete Fourier transform of $\underline{F}_T(j)$:

$$\underline{f}_T(\tau) = \underline{c}_0 + \sum_i \underline{c}_i \cos(\omega_i \tau + \phi_i) + m\underline{g}, \quad 0 \le \tau \le 1 \quad (2)$$

where τ is normalized time $\left(\frac{t-t_0}{\Delta t}\right)$ and $\omega_i = \frac{2\pi j_i}{\Delta t}$ for some integer j_i from 0 to $N-1$. Applied ground contact force $\underline{f}(\tau)$ is:

$$\underline{f}(\tau) = \underline{c}_0 + \sum_i \underline{c}_i \cos(\omega_i \tau + \phi_i) \quad (3)$$

This representation can be easily integrated to obtain expected positions and velocities. Orientation and angular velocity can be estimated from a rigid body simulation.

4 Altering a Stance (Dynamic)

To drive a character from a fixed set of examples, we need some way to fill the gaps in the state space. The technique proposed here is to edit stances by scaling applied forces:

$$\underline{\tilde{f}}(\tau) = \underline{\beta}\ \underline{f}(\tau) \quad (4)$$

where scale factor $\underline{\beta}$ is a square, diagonal matrix with elements β_v, β_h, and β_p (scale factors in the *vertical*, *heading*, and *perpendicular* directions).

Equation 4 is really a definition of similarity – a stance is similar to the original if it can be obtained by scaling applied forces. Given this definition of similarity, five parameters can be adjusted to obtain a new stance: β_v, β_h, β_p, and landing and launch times t_{TD} and t_{LO}. Desired position and velocity (in the form of parameters zs, vs, ze, ve, vp, and x) fully determine the five unknown scale and time parameters. A derivation can be found in Appendix A. This technique of scaling forces is interesting because it combines a physically-based representation of motion (however simple) with an analytical technique for editing that motion to meet specific position and velocity constraints.

4.1 Limits on Stance Transformation

Scaling forces by arbitrary scale factors will produce many results that are physically implausible. These results are filtered using four types of constraints, described in equations 5 through 8. Scale factors must be positive:

$$\beta_i \ge 0, \quad i \in \{v, h, p\} \quad (5)$$

The maximum force is limited to a value (F_{max}) obtained by examining forces in the database:

$$\beta_h^2 f_h^2(\tau) + \beta_p^2 f_p^2(\tau) + \beta_v^2 f_v^2(\tau) < F_{max}^2, \quad 0 \le \tau \le 1 \quad (6)$$

Forces must remain within a given friction cone (with coefficient of friction μ) over the entire stance:

$$\beta_h^2 f_h^2(\tau) + \beta_p^2 f_p^2(\tau) \le \mu^2 \beta_v^2 f_v^2(\tau), \quad 0 \le \tau \le 1 \quad (7)$$

(Equation 7 assumes level ground – the ground contact normal is in the vertical direction.)

Finally, the distance from the contact point to the center of mass must remain within a given range for the entire stance:

$$\chi_{min}^2 \le \|\underline{x}(\underline{\beta}, \tau) - \underline{x}_{cp}(\tau)\|^2 \le \chi_{max}^2, \quad 0 \le \tau \le 1 \quad (8)$$

where $\underline{x}(\underline{\beta}, \tau)$ is position of the center of mass, $\underline{x}_{cp}(\tau)$ is the position of the contact point, and χ_{min} and χ_{max} are limits on allowable distance from contact point to center of mass, set based on the kinematics of the character.

As written, equations 6 through 8 would be tested by sampling τ from 0 to 1 for each candidate set of scale factors $\underline{\beta}$ and contact point $\underline{x}_{cp}(\tau)$. We can do better, however. For any τ, equation 6 represents an axis-aligned ellipsoid centered at the origin. This constraint is approximated with a single ellipsoid contained within constraint ellipsoids for all τ. Offline computation of coefficients:

$$A_i = \max_\tau f_i^2(\tau), \ i \in \{v, h, p\} \quad (9)$$

allows us to do a simple check at runtime:

$$A_h \beta_h^2 + A_p \beta_p^2 + A_v \beta_v^2 < F_{max}^2 \quad (10)$$

For any τ, equation 7 represents an axis-aligned ellipse in the $\frac{\beta_h}{\beta_v}$, $\frac{\beta_p}{\beta_v}$ plane, centered at the origin. This constraint

is approximated in a similar fashion. Offline computation of coefficients:

$$A_h = \max_\tau \left(\frac{f_h(\tau)}{f_v(\tau)} \right)^2 \quad A_p = \max_\tau \left(\frac{f_p(\tau)}{f_v(\tau)} \right)^2 \tag{11}$$

allows the following simple runtime check:

$$A_h \left(\frac{\beta_h}{\beta_v} \right)^2 + A_p \left(\frac{\beta_p}{\beta_v} \right)^2 \leq \mu^2 \tag{12}$$

For any τ, equation 8 expresses an axis aligned ellipsoid that is not centered at the origin. Nevertheless, an approximation to this constraint can be created in a manner similar to that for equation 7. (See Appendix B for a derivation.) We assume that contact position $\underline{x}_{cp}(\tau)$, expressed relative to the center of mass at landing, is known. Because ellipsoid center is a function of τ, the approximation is not conservative, but it was quite good for the motions we tested (e.g. see Figure 5).

4.2 Scaling to New Characters

Scaling stances to new characters is very straightforward in a force based scheme. The technique we use is based on the concept of dynamic similarity. Although dynamically similar motion cannot be obtained for two different characters in general, it can be obtained for the rigid body approximation used here. Time, state, and forces for a stance are scaled based on a relative length measurement L and a relative mass measurement M. Scaling rules are summarized in [17] and [7].

Direct scaling is appealing, because it is simple and fast, and it preserves the dynamic behavior of the simulated motion. It also maintains the constraints expressed in equations 5 through 8. All forces are scaled by the same positive scale factor (M), and all positions by the same positive scale factor (L). Therefore, constraints are maintained trivially if we can assume that the coefficient of friction remains the same, the minimum and maximum extension scale linearly with L and the maximum force F_{max} scales linearly with M, all of which are plausible for humanlike figures. In fact, relative extension limits may provide a good way to obtain the length scale factor L.

5 Full Character Motion (Kinematic)

Although we begin with reference motion for humanlike characters, motion editing is performed using a rigid body approximation. A rigid body simulator allows forces and center of mass motion to be visualized, but motion of a more complex character is the desired end product for animation.

$speed(m/s)$	β_h	μ_{min}	ext_{max}	cp_h
0.0	0.00	0.11	1.12	0
2.0	0.43	0.23	1.12	0.18
4.0	0.86	0.42	1.12	0.36
6.0	1.29	0.62	1.26	0.53

Table 1: Forces extracted from the example run can be scaled to achieve steady-state velocities from 0 to $6m/s$. (The original run was $4.6m/s$.) Parameters are force scale factor in the *heading* direction (β_h), the coefficient of friction required (μ_{min}), maximum distance from center of mass to contact point (ext_{max}), and position of the contact point in the *heading* direction, expressed relative to the center of mass at landing (cp_h).

We use a simple, kinematic process to create animated motions such as that in Figure 1 from a rigid body simulation. Our goals are to ensure that the foot of the character stays planted and that the center of mass and orientation of the character follow the intended trajectories.

In a first pass, each stance is considered separately. The character is rooted at the contact point, and joint angle trajectories are found which minimize error in center of mass position, error in character orientation, and deviation from the reference motion. In a second pass, the motion segments computed for individual stances are blended during the flight phase connecting those stances to create continuous, smooth motion.

6 Examples

Figure 1 shows results from two source motions: a simulated run and a keyframed jump. These example motions were edited to change running velocity, to scale the run to a new character, and to illustrate the range of jumps that can be created by scaling the single example jump.

6.1 Running Velocities

A running stance is a period of continuous left or right foot contact, followed by a period of flight. A total of 11 sequential stances were extracted from the example data. Because the example motion was simulated, each stance was slightly different. To create a steady-state run cycle, we chose a pair of right and left stances that fit closely and scaled them slightly to create seamless transitions.

The resulting run cycle was then scaled to new steady state velocities ranging from 0 to $6m/s$. Table 1 shows some parameters from those experiments. All of the run cycles were viable given a coefficient of friction of 0.7. The $6m/s$ run is near the kinematic limits of the character.

Because the stances extracted from the example motion contain some accelerating and some decelerating stances,

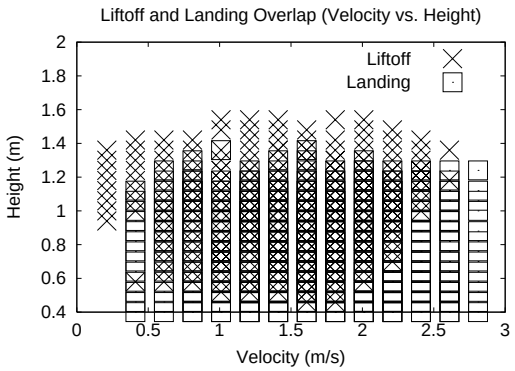

Figure 3: For the jump, stances must be scaled so that ve and ze at liftoff match vs and zs for landing. This plot shows the range of overlap (velocity vs. height).

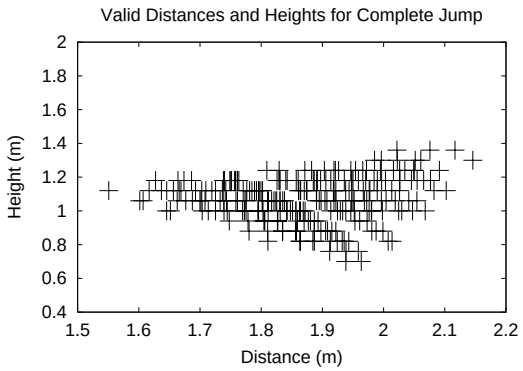

Figure 4: Jumps (distance vs. height) that can be created by scaling forces extracted from the example motion. No point lies more than 5% beyond the extremes of force, friction, and extension from the original motion.

it is possible to plan motions where speed is dynamically modulated. Simulating extended motion sequences, however, is beyond the scope of this paper, because we have not described a general technique for controlling orientation. We can easily plan a sequence of stances to decelerate the runner by $0.4m/s$ per stride, for example, but after 5-6 strides, the runner's orientation would begin to diverge from its desired value. Our current system does allow orientation to be partially controlled by controlling contact placement, but much better solutions are certainly possible. We are currently adding a simple controller to the system, for example, so that longer, dynamically changing sequences of stances can be explored.

6.2 Scaling to a New Character

To create a running sequence for a troll, we scaled the steady-state run cycle from the man to the troll using the rules summarized in [17] and [7]. Relative mass M was 3.62, and relative length L was 0.67. For this example, the choice of a single length scale factor was somewhat limiting – the troll is shorter than the man, but he is also wider. Our first attempt resulted in foot placement much too close to the center of the body (see Figure 1). This problem can be controlled by scaling parameter vp. Scaling perpendicular velocity causes the optimal contact point to move laterally to stabilize orientation.

6.3 Scaling Jump Distances and Heights

Two stances were extracted from the example jump – one for liftoff and one for landing. To find jumps of new heights and distances, we must find scale factors that blend the liftoff motion seamlessly into the landing. Fortunately, the stance parameters we use make this easy – ve and ze for liftoff must match vs and zs for landing. Figure 3 shows a

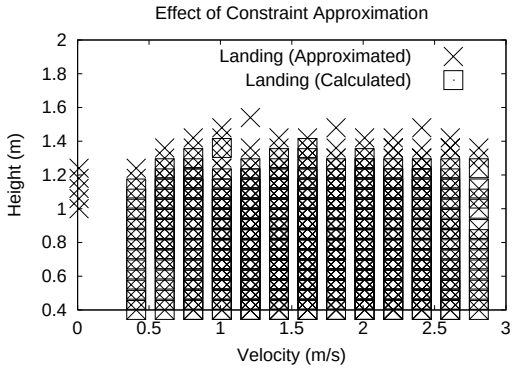

Figure 5: The constraint approximations create a state space very similar to the original.

sampling of velocity in the heading direction vs. height at the apex of the flight phase between the two stances. Any point in the overlapping region is a candidate for forming a new jump. Figure 4 shows the range of jumps that can be created by scaling the original example. All of these jumps have a maximum force, a maximum required friction coefficient, and maximum and minimum extensions no more than 5% beyond the bounds of the original motion. Animations of some of these jumps, along with other results from this paper can be viewed from the web pages at: http://www.cs.brown.edu/people/nsp/.

Figure 5 illustrates the effect of using the constraint approximations described in Section 4.1. Because the extension approximation is not conservative, some additional points appear in the approximation. Overall, however, the approximation is quite good.

7 Discussion

This paper has described a technique for editing example motion for locomotion tasks. Locomotion tasks are represented in terms of applied forces required to move the character, and forces are altered to achieve new goals, such as dealing with an increased load, changing velocities, or altering the height and distance of a jump. Basic frictional and kinematic constraints are maintained for a rigid body approximation of the character. This editing technique is fast – the only search or sampling performed to create a new rigid body simulation is a two-dimensional search for a contact point (to help stabilize orientation).

The algorithm described in this paper is not a control algorithm, and there is much to be done to make it appropriate for use with an actual robot. We are currently investigating the use of force control with computed force as a target. Additional feedback is incorporated to keep the center of mass in the correct position and to maintain balance and posture. Other areas of future work include incorporating biomechanical constraints beyond the simple kinematic check discussed here and planning extended motions from a much larger library of stances. We hope that this work will ultimately contribute to the development of new techniques for example driven robotic motion.

Acknowledgments

The authors wish to thank Jessica Hodgins for providing the simulated running motion. Thanks also to Remco Chang, who wrote the software to map rigid body motion to an animation of the full character, and Carolyn Uy, who developed the rigid body simulator for stance sequences.

References

[1] ASADA, H., AND ASARI, Y. The direct teaching of tool manipulation skills via the impedance identification of human motion. In *Proc. IEEE Intl. Conference on Robotics and Automation* (1988).

[2] ATKESON, C. G., MOORE, A. W., AND SCHAAL, S. Locally weighted learning for control. *Artificial Intelligence Review 11* (1997), 75–113.

[3] DEMPSTER, W. T., AND GAUGHRAN, G. R. L. Properties of body segments based on size and weight. *American Journal of Anatomy 120* (1965), 33–54.

[4] DILLMANN, R., FRIEDRICH, H., KAISER, M., AND UDE, A. Integration of symbolic and subsymbolic learning to support robot programming by human demonstration. In *Robotics Research: The Seventh International Symposium*, G. Giralt and G. Hirzinger, Eds. Springer, New York, 1996.

[5] GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. Neuroanimator: Fast neural network emulation and control of physics-based models. In *SIGGRAPH 98 Proceedings* (1998), Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 9–20.

[6] HARAI, K., HIROSE, M., HAIKAWA, Y., AND TAKENAKA, T. The development of the honda humanoid robot. In *Proc. IEEE Intl. Conference on Robotics and Automation* (1998).

[7] HODGINS, J. K., AND POLLARD, N. S. Adapting simulated behaviors for new characters. In *SIGGRAPH 97 Proceedings* (1997), ACM SIGGRAPH, Addison Wesley, pp. 153–162.

[8] HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. Animating human athletics. In *SIGGRAPH 95 Proceedings* (Aug. 1995), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 71–78.

[9] KANG, S. B., AND IKEUCHI, K. Toward automatic robot instruction from perception – temporal segmentation of tasks from human hand motion. *IEEE Transactions on Robotics and Automation 11*, 5 (1995), 670–681.

[10] KODITSCHEK, D. E., AND BÜHLER, M. Analysis of a simplified hopping robot. *International Journal of Robotics Research 10*, 6 (December 1991), 587–605.

[11] LASZLO, J., VAN DE PANNE, M., AND FIUME, E. Limit cycle control and its application to the animation of balancing and walking. In *SIGGRAPH 96 Proceedings* (Aug. 1996), Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 155–162.

[12] MCMAHON, T. A., AND CHENG, G. C. The mechanics of running: how does stiffness couple with speed? *Journal of Biomechanics 23* (1990), 65–78.

[13] PANDY, M. G., ANDERSON, F. C., AND HULL, D. G. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Transactions of the ASME 114* (nov 1992), 450–460.

[14] POPOVIĆ, Z., AND WITKIN, A. Physically-based motion transformation. In *SIGGRAPH 99 Proceedings* (Aug. 1999), Annual Conference Series, ACM SIGGRAPH, ACM Press.

[15] PRATT, J., AND PRATT, G. Intuitive control of a planar bipedal walking robot. In *Proc. IEEE Intl. Conference on Robotics and Automation* (Leuven, Belgium, 1998).

[16] RAIBERT, M. H. *Legged robots that balance*. MIT Press, 1986.

[17] RAIBERT, M. H., AND HODGINS, J. K. Animation of dynamic legged locomotion. In *Computer Graphics (SIGGRAPH 91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 349–358.

[18] ROSE, C. F., COHEN, M. F., AND BODENHEIMER, B. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications September/October* (1998), 32–40.

[19] SARANLI, U., SCHWIND, W. J., AND KODITSCHEK, D. E. Toward the control of a multi-jointed, monoped runner. In *Proc. IEEE Intl. Conference on Robotics and Automation* (1998).

[20] WITKIN, A., AND KASS, M. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Proceedings)* (Aug. 1988), J. Dill, Ed., vol. 22, pp. 159–168.

Appendix A: Stance Scaling Derivation

Let stance times be 0 the start of the stance, t_{TD} at landing, t_{LO} at liftoff, and t_E at the end of the stance. Parameters t_{TD}, t_{LO}, and t_E are unknown, but time on the ground (Δt) is fixed for a stance, and so:

$$t_{TD} = t_{LO} - \Delta t \tag{13}$$

Rewrite force equation 2 in terms of actual time t instead of normalized time τ and incorporate scale factors:

$$\underline{\tilde{f}}_T(t) = \underline{\beta}\ \underline{c}_0 + \sum_i \underline{\beta}\ \underline{c}_i \cos(\tilde{\omega}_i t + \tilde{\phi}_i) + m\underline{g}, \quad t_{TD} \le t \le t_{LO} \tag{14}$$

Integrate equation 14 from time t_{TD} to t_{LO} and divide by mass m to obtain an expression for change in velocity:

$$\dot{\underline{x}}(t_{LO}) - \dot{\underline{x}}(t_{TD}) = \frac{\underline{\beta}\ \underline{c}_0 \Delta t}{m} + \underline{g}\Delta t \tag{15}$$

Add the ballistic segments of the stance:

$$\dot{\underline{x}}(t_E) - \dot{\underline{x}}(0) = \frac{\underline{\beta}\ \underline{c}_0 \Delta t}{m} + \underline{g} t_E \tag{16}$$

In terms of stance parameters:

$$ve - vs = \frac{\beta_h\ c_{0,h} \Delta t}{m} \tag{17}$$

$$vp = \frac{\beta_p\ c_{0,p} \Delta t}{m} \tag{18}$$

$$gt_E = \frac{\beta_v\ c_{0,v} \Delta t}{m} \tag{19}$$

where g is $9.8 m/s$.

Integrate velocities from one apex to the next to obtain an expression for change in position:

$$\underline{x}(t_E) - \underline{x}(0) = \tag{20}$$
$$\dot{\underline{x}}(0)\ t_{LO} + \dot{\underline{x}}(t_E)(t_E - t_{LO}) + \underline{g} t_E \left(t_{LO} - \frac{t_E}{2}\right) + \frac{\underline{\beta}\ \underline{c}_0 \Delta t^2}{2m} - \sum_i \frac{\underline{\beta}\ \underline{c}_i \Delta t^2}{m\omega_i} sin(\phi_i)$$

In terms of stance parameters:

$$x - ve\ (t_E - t_{LO}) - vs\ t_{LO} = \tag{21}$$
$$\frac{\beta_h c_{0,h} \Delta t^2}{2m} - \sum_i \frac{\beta_h c_{i,h} \Delta t^2}{m\omega_i} sin(\phi_i)$$

$$ze - zs + gt_E \left(t_{LO} - \frac{t_E}{2}\right) = \tag{22}$$
$$\frac{\beta_v c_{0,v} \Delta t^2}{2m} - \sum_i \frac{\beta_v c_{i,v} \Delta t^2}{m\omega_i} sin(\phi_i)$$

The six equations 13, 17, 18, 19, 21, and 22 can be solved algebraically for t_{TD}, t_{LO}, t_E, β_h, β_v, and β_p.

Appendix B: Extension Constraint

Expression $\underline{x}(\underline{\beta}, \tau)$ is obtained by integrating $\underline{f}_T(\tau) = m\ddot{\underline{x}}(\tau)$ using equation 2 for $\underline{f}_T(\tau)$. This integration results in an expression containing a scaled and an unscaled portion:

$$\underline{x}(\underline{\beta}, \tau) = \underline{x}_c(\tau) + \underline{\beta}\ \underline{x}_s(\tau) \tag{23}$$

We assume contact point $\underline{x}_{cp}(\tau)$ is known. It does not scale with $\underline{\beta}$ and can be folded into the $\underline{x}_c(\tau)$ term. With a little manipulation, equation 8 can be put into the following form:

$$\chi^2_{min} \le \left(\frac{\beta_h - a_h(\tau)}{b_h(\tau)}\right)^2 + \left(\frac{\beta_p - a_p(\tau)}{b_p(\tau)}\right)^2 + \left(\frac{\beta_v - a_v(\tau)}{b_v(\tau)}\right)^2 \le \chi^2_{max} \tag{24}$$

For any τ, the constraint boundaries of equation 24 are axis-aligned ellipsoids in β_v, β_h, β_p space, centered at $\underline{a}(\tau)$. To approximate the χ_{max} constraint, we compute an ellipsoid that fits within the axis-aligned bounding box contained within the bounding boxes of ellipsoids at all τ. (This is not a conservative approximation.)

$$\underline{R}_{max} = \min_\tau \left(\underline{a}(\tau) + \|\underline{b}(\tau)\ \chi_{max}\|\right) \tag{25}$$

$$\underline{R}_{min} = \max_\tau \left(\underline{a}(\tau) - \|\underline{b}(\tau)\ \chi_{max}\|\right) \tag{26}$$

$$\underline{a}_r = \frac{\underline{R}_{max} + \underline{R}_{min}}{2}, \quad \underline{b}_r = \frac{\underline{R}_{max} - \underline{R}_{min}}{2\chi_{max}} \tag{27}$$

$$\left(\frac{\beta_h - a_{r,h}}{b_{r,h}}\right)^2 + \left(\frac{\beta_p - a_{r,p}}{b_{r,p}}\right)^2 + \left(\frac{\beta_v - a_{r,v}}{b_{r,v}}\right)^2 \le \chi^2_{max} \tag{28}$$

A runtime test is then performed using equation 28, where $\underline{a}_r$ and $\underline{b}_r$ are constants.

Similarly, for the χ_{min} constraint, we compute the ellipsoid that is defined by the bounding box that contains the bounding boxes of the ellipsoids at all τ:

$$\underline{S}_{max} = \max_\tau \left(\underline{a}(\tau) + \|\underline{b}(\tau)\ \chi_{min}\|\right) \tag{29}$$

$$\underline{S}_{min} = \min_\tau \left(\underline{a}(\tau) - \|\underline{b}(\tau)\ \chi_{min}\|\right) \tag{30}$$

$$\underline{a}_s = \frac{\underline{S}_{max} + \underline{S}_{min}}{2}, \quad \underline{b}_s = \frac{\underline{S}_{max} - \underline{S}_{min}}{2\chi_{min}} \tag{31}$$

$$\chi^2_{min} \le \left(\frac{\beta_h - a_{s,h}}{b_{s,h}}\right)^2 + \left(\frac{\beta_p - a_{s,p}}{b_{s,p}}\right)^2 + \left(\frac{\beta_v - a_{s,v}}{b_{s,v}}\right)^2 \tag{32}$$

A runtime test is performed using equation 32, where $\underline{a}_s$ and $\underline{b}_s$ are constants.