EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Feedback Control for Rotational Movements in Feature Space

M. Al Borno[1], E. Fiume[1], A. Hertzmann[2] and M. de Lasa[3]

[1]University of Toronto
[2]Adobe Systems
[3]Autodesk Canada

Abstract

Synthesizing controllers for rotational movements in feature space is an open research problem and is particularly challenging because of the need to precisely regulate the character's global orientation, angular momentum and inertia. This paper presents feature-based controllers for a wide variety of rotational movements, including cartwheels, dives and flips. We show that the controllers can be made robust to large external disturbances by using a time-invariant control scheme. The generality of the control laws is demonstrated by providing examples of the flip controller with different apexes, the diving controller with different heights and styles, the cartwheel controller with different speeds and straddle widths, etc. The controllers do not rely on any input motion or offline optimization.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Rotation is a key component of many of the most impressive movements of ballet performers, breakdancers and gymnasts. Rotational movements have not yet been thoroughly investigated in the character control literature. The problem is particularly challenging because of the need to precisely regulate the character's global orientation, angular momentum and inertia. Most existing methods rely on motion capture or pre-scripted animations, which limits their generality, or require substantial manual effort. Recently, some researchers have designed control algorithms for landing and rolling movements as a first step towards the study of rotational dynamics. Our work constitutes a further step towards this goal.

This paper presents feature-based controllers for a wide variety of rotational movements, including cartwheels, dives and flips. Most of the rotations are planar, but the controllers are fully three-dimensional. Our control laws are intuitive to design and general, which we demonstrate by providing examples of the flip controller with different apexes, the diving controller with different heights and styles, the cartwheel controller with different speeds and straddle widths, etc. The control strategies are often used across different types of movements. For instance, almost the same controller that makes a character land on its feet in a backflip also allows a diver to enter the water in a fully extended and straight posture. The identical controller can be applied to characters with very different body proportions. Our controllers do not rely on any input motion or offline optimization, and run at interactive rates. However, some effort is required to synthesize the controllers since they are hand-tuned.

We place a particular emphasis on the robustness of the controllers. As is well-known, time-invariant (or state-based) controllers are more robust to disturbances than time-based controllers since they do not attempt to adhere to the the timing of a pre-defined motion. In this work, we show that controllers for rotational movements can be made robust by substituting time with a phase variable that positions the character in the revolution.

2. Related work

The main focus of previous research in physics-based character animation has been on balancing [AdSP07, MZS09]

and locomotion [MdLH10, WFH09, WP09, WP10, YLv07]. In this section, we discuss some of the work that touched upon rotational movements, either in the context of trajectory optimization or controller design.

Trajectory optimization synthesizes motion by solving a constrained nonlinear optimization problem [WK88]. The time required to solve the optimization problem restrains the method to offline applications. Safonova et al. [SHP04] and Fang et al. [FP03] used motion capture and pre-scripted animations to facilitate the optimization of flips and leaps. However, their approach can only generate motions that are near the input motions. Liu et al. [LYvdP*10] and Al Borno et al. [ABdLH13] developed sampling methods to synthesize contact-rich and highly dynamics motions such as rolls and flips. The problem is that trajectory optimization merely synthesizes an open-loop trajectory. One can use feedback around the trajectory to obtain a controller [dSAP08, LYvdPG12, MLPP09, YL10], but it falls short of being general. For instance, trajectory optimization can be used to generate a flip with a certain apex. Generating a flip with a different apex requires a new optimization, which takes several minutes.

Controllers are better suited than trajectory optimization for real-time applications. The two general methods to design controllers are joint space control and feature-based control (also known as task space control). One can synthesize controllers for specific actions such as balancing, walking and jumping, or to track a reference motion [AdSP07, dSAP08, JYL09, LKL10, WHDK12].

Manually designing controllers in joint space can be very difficult and time-consumming due to the nonlinear interaction between individual joints. Offline optimization eases the process [WFH09, ASvdP13], but manual specification of the controller's structure is still needed. Once a controller is synthesized, it is usually specific for a character [HP97], and changing the properties of the movement is unintuitive or requires a new optimization. In important early work, Hodgins et al. [HWBO95] and Wooten et al. [WH96, Woo98] synthesized dives, flips and handsprings in joint space. However, the controllers are too specific because motion properties such as the style, the speed, the position of the apex of the flip, the height of the dive, etc., cannot be modified without re-tuning the parameters. Recently, Sehoon et al. [HYL12] developed a control strategy for landing and rolling movements based on the optimization of an abstract model. In contrast, we found that control laws based on simple mechanics could generate movements involving an airborne stage such as flips and dives.

Feature-based controllers offer an abstraction layer to individual joints, where control is specified in terms of high level features of the motion, such as the trajectory of the center-of-mass [AdSP07]. Feature-based controllers are usually more intuitive to design and more robust to changes to the shape of the character than joint space controllers. Previous researchers have used feature-based controllers to generate balance [AdSP07, MZS09], walking [dLMH10, WP10, WZ10], jumping [dLMH10], and recently rolling movements [BMYZ13], but have not yet been able to generate highly dynamic movements such as cartwheels and flips [BMYZ13, dLMH10]. In this paper, we present feature-based controllers for a wide variety of rotational movements. We show that the controllers succeed for a large range of inputs, which demonstrate their generality.

In the control systems literature, virtual constraints have become an important tool to design time-invariant controllers for various problems [MC13], including bipedal locomotion [AP11, Ame13, GCAS10]. Let $\mathbf{q}$ be the vector of generalized joint positions of a mechanical system. The method of virtual constraints expresses the controlled variables $\mathbf{q}_i$ as a function of a phase variable $\theta(\mathbf{q})$:

$$\mathbf{q}_i = h_i(\theta), \tag{1}$$

instead of expressing them as a function of time $\mathbf{q}_i(t)$. This is always possible when the phase variable is strictly monotonic with respect to time. Walking controllers often take θ to be the angle of the line between the stance leg end and the hip [AP11, GCAS10]. We draw inspiration on this literature to design time-invariant controllers for rotational movements.

3. Preliminaries

We now provide an overview of feature-based control [AdSP07, dLMH10, Lie77]. Let $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ denote generalized joint positions, velocities and accelerations. A *feature* is the output of a map of the character's state:

$$\mathbf{y} = f(\mathbf{q}, \dot{\mathbf{q}}). \tag{2}$$

Examples of features are the center-of-mass (COM), the angular momentum and an end-effector.

The objectives used in this paper are $E_{(l/r)footContact}$ and $E_{(l/r)handContact}$ to keep the (left/right) foot and the (left/right) hand planted to the ground, $E_{(l/r)foot}$ and $E_{(l/r)hand}$ to control the trajectory of the (left/right) foot and (left/right) hand, E_{COM} to control the trajectory of the COM, E_{AM} to control the angular momentum, and E_{pose} to servo full-body joints to a rest pose.

Let $\bar{\mathbf{y}}_i$ refer to the desired feature associated with objective E_i. The objectives measure the difference between the desired and actual feature accelerations:

$$E_i = \|\bar{\ddot{\mathbf{y}}}_i - \ddot{\mathbf{y}}_i\|^2. \tag{3}$$

The E_{AM} objective, however, measures the difference between the desired and actual angular momentum velocities [dLMH10]. In this work, the desired feature accelerations are computed by linear control:

$$\bar{\ddot{\mathbf{y}}}_i = k_p(\bar{\mathbf{y}}_i - \mathbf{y}_i) - k_v\dot{\mathbf{y}}_i, \tag{4}$$

where k_p and k_v are gains.

As in previous work [AdSP07], we use a two-layer control architecture hierarchy. At the higher level, a control strategy determines the desired features. At the lower-level, a quadratic program is solved to map features to joint torques.

Let $\mathbf{x} = [\tau^T, \ddot{\mathbf{q}}^T, \zeta^T]^T$, where τ denotes joint torques and ζ denotes the basis weights of the linearization of the Coulomb friction cone [AdSP07]. The following quadratic program is solved at each simulation timestep:

$$\mathbf{x} = \underset{x}{\operatorname{argmin}} \sum_i w_i E_i, \tag{5}$$

$$\mathbf{C}(\mathbf{x}) = 0, \mathbf{D}(\mathbf{x}) \geq 0, \tag{6}$$

where the equality constraints $\mathbf{C}$ are the equations of motion, and the inequality constraints $\mathbf{D}$ account for contact forces, joint limits and torque limits [dLMH10]. Given τ, a simulator updates the character's state by integration.

4. Phase parametrization

One can create feature-based controllers by specifying the desired features as functions of time $\bar{\mathbf{y}}(t)$. This is the approach taken by most of the literature [AdSP07, dSAP08, dLMH10, MZS09, WP10]. One exception is Brown et al. [BMYZ13], where it is shown how to design robust rolling controllers that track a reference motion independently of time. In this paper, we extend the work of Brown et al. [BMYZ13] by designing time-invariant control schemes for a variety of rotational movements, without using reference motions. Previous time-invariant controllers were mostly restricted to locomotion and to joint space [AP11, GCAS10]. To achieve our goal, we use the method of virtual constraints (1), with a small modification to make it applicable in feature space. Specifically, we specify the desired features as functions of a phase variable $\theta(\mathbf{q})$ and the character's state:

$$\bar{\mathbf{y}}_i = h_i(\theta, \mathbf{q}, \dot{\mathbf{q}}). \tag{7}$$

It is redundant to include θ in $h_i(\cdot)$ since it is a function of $\mathbf{q}$, but we do so to emphasize the importance of the phase parametrization in our control design. When presenting the details of the controllers, we will see that it is easy to define $h_i(\cdot)$ by hand, although optimization might be necessary for more complex motions.

In the remainder of this section, we discuss our choice of a phase variable and the benefits of the parametrization. The phase variable needs to be strictly monotonic in time in the undisturbed trajectory so that it can be used to uniquely position the character. An natural choice for θ in rotational movements is an angle in the plane perpendicular to the axis of rotation of the revolution. For rotations in the sagittal plane such as flips and cartwheels, we chose θ to be the angle of the line between the neck and the pelvis (see Fig. 1). There are many other possible choices. In this paper, we use the convention that θ increases as the revolution progresses. A trajectory can be partitioned in domains when it does not have a variable that is strictly monotonic for its entire duration. For example, movements that involve multiple revolutions such as dives are partitioned by revolution, so that θ is strictly monotonic on each domain.

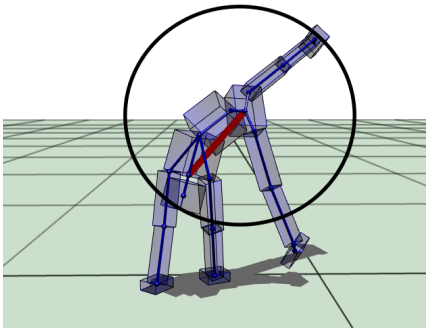

Figure 1: ***Controllers for rotational movements parametrized by phase*** *We show that parametrizing controllers for rotational movements by a phase variable can make them general and robust. The variable is used to uniquely position the character in the trajectory instead of time. A natural choice of a phase variable for rotational motion is an angle in the plane of rotation. In this image, the angle of the line between the neck and the pelvis is the phase variable.*

Parametrizing controllers by a phase variable offers two benefits. The first is to make the controllers more general. For example in a flip, the character roughly completes a full revolution, no matter how far, high or fast the character jumps. Generality is achieved by using this invariant property of the movement in the control structure. The second is to make the controllers more robust because of feedback, which we illustrate with the following example. In Fig. 2, we show the character performing a cartwheel under no disturbances. Now, assume that strong wind forces push the character in the direction of travel, so that the character is performing the cartwheel faster than usual. If the desired features $\bar{\mathbf{y}}$ are parametrized by time, then the character is likely to go out of phase since it is ahead of schedule, but $\bar{\mathbf{y}}$ remains on schedule. This does not occur when the desired features $\bar{\mathbf{y}}$ are parametrized by θ. The reason is that when the character gets disturbed, θ gets disturbed. The controller uses θ to position the character in the revolution and use the corresponding features.

5. Rotational movements

In this section, we present our approach to the synthesis of physics-based rotational movements. We emphasize that this work is about designing general controllers, not just trajectories. We will see that the design of complex rotational motions can be decomposed in stages that, in humans, correspond to relatively stable configurations that can be practiced and rehearsed. For all the controllers in this work, the E_{pose} objective is active during the entire movement. The initial value of $\bar{\mathbf{y}}_{pose}$ is set to be the desired initial pose of the movement. This value is maintained during the entire movement, unless we mention changes in specific joints, similarly to Wu et al. [WZ10]. Given a vector $\mathbf{v}$, we denote its projection on the ground plane by $\mathbf{v}^{xz}$ and on the vertical axis by $\mathbf{v}^{y}$.

5.1. Cartwheel

In this section, we describe the left side cartwheel controller. For the right side cartwheel, replace the left hand/foot with the right hand/foot, and vice versa. The cartwheel controller consists of 5 stages, with θ used to determine the current stage of the movement (see Fig. 2). Stage i is triggered when $\theta \in (l_i, u_i)$, where θ increases as the cartwheel progresses. The controller has the following structure:

$$\bar{\mathbf{y}}_i = \lambda_i(\mathbf{q}, \dot{\mathbf{q}}), \quad \theta \in (l_i, u_i), \tag{8}$$

where λ_i is the control law associated with stage i. The controller takes as input the desired straddle width, the desired distance between the hands when in contact (s), and a speed factor (v).

Let $\mathbf{d}$ be the desired direction of the cartwheel. During the entire movement, an angular momentum E_{AM} objective with $\bar{\mathbf{y}}_{AM}$ in the direction perpendicular to $\mathbf{d}$ is used. We control the speed of the cartwheel with the magnitude of $\bar{\mathbf{y}}_{AM}$:

$$|\bar{\mathbf{y}}_{AM}| = v|\tilde{\mathbf{y}}_{AM}|, \tag{9}$$

where $|\tilde{\mathbf{y}}_{AM}|$ is the magnitude of the angular momentum target at a nominal speed. Setting $v = 0.5$ and $v = 1.5$ generate cartwheels that are approximately 50% slower and 50% faster than the nominal speed, respectively.

On stage 1, the rest pose of the lumbar joint is chosen so that the character tilts towards its left side. An E_{COM} objective is used for the character to bend down and move in the $\mathbf{d}$ direction:

$$\bar{\mathbf{y}}_{COM} = \mathbf{c} + \eta\mathbf{d} + [0, l, 0]^T, \tag{10}$$

where $\mathbf{c}$ is the position of the COM, and η and l are scalars that satisfy $\eta > 0$ and $l < 0$. This objective remains active until both hands touch the ground. The feet are maintained on the ground with the $E_{rfootContact}$ and $E_{lfootContact}$ objectives. On stages 2 and 3, the E_{lhand} objective is used to place the character's left hand:

$$\bar{\mathbf{y}}^{xz}_{lhand} = \mathbf{c}^{xz} + \eta\mathbf{d}^{xz} \tag{11}$$

$$\bar{\mathbf{y}}^{y}_{lhand} = \mathbf{y}^{y}_{lhand}(1 - \frac{\theta - \theta_i}{\theta_f - \theta_i}), \tag{12}$$

where θ_i is the value of θ when the character should begin to lower its hand, and θ_f is the value of θ when the hand should be on the ground. The same approach is used to place the right hand on stage 3, except that the target vertical position goes to 0 as the distance between the hands gets closer to its desired value:

$$\bar{\mathbf{y}}^{y}_{rhand} = \mathbf{y}^{y}_{rhand}(1 - \frac{||\mathbf{y}^{xz}_{rhand} - \mathbf{y}^{xz}_{lhand}||_2}{s}). \tag{13}$$

The hand contact objectives $E_{handContact}$ are activated when the hands are sufficiently close to the ground. The rest poses of the hips are used to create the desired straddle width. As the character gets near the end of its revolution, on stages 4 and 5, respectively, the $E_{lhandContact}$ and the $E_{rhandContact}$ objectives are deactivated. The E_{lfoot} and E_{rfoot} objectives are used so that the character can land its feet in a good position. We set $\bar{\mathbf{y}}_{lfoot}$ and $\bar{\mathbf{y}}_{rfoot}$ relative to the COM and ϕ, which is an estimate of the character's global orientation. Specifically, we have:

$$\bar{\mathbf{y}}_{lfoot} = R_\phi(\mathbf{c}^{xz} + \mathbf{p}^{loff}), \tag{14}$$

where R_ϕ is the rotation matrix associated with ϕ, and $\mathbf{p}^{loff}$ is a fixed horizontal offset term for the left foot. An analogous feedback law applies for $\mathbf{y}_{rfoot}$. We found that the orientation of the lumbar joint is a good choice for ϕ. When the left and right foot are close enough to the ground, we activate the $E_{lfootContact}$ and the $E_{rfootContact}$ terms. The movement ends with a balance controller, which is similar to what was presented in Abe et al. [AdSP07] and Kudoh et al. [KKI06].

As illustrated in the accompanying video, the controller is robust to pushes that speed up or slow down the movement since it is time-invariant. The controller also exhibits robustness to pushes in random directions because the positions of the end-effectors are not specified in world coordinates, but as functions of the COM and the character's orientation (see (11) and (14)). We also give different examples of cartwheels by varying the inputs, which clearly demonstrates that we are synthesizing control solutions that are more general than a single trajectory.

5.2. Flips

In this section, we provide the details of the backflip controller. We can generate frontflips using the same approach (see the accompanying video). The main difference between the backflip and frontflip controller is the sign of the angular momentum target $\bar{y}_{AM}$ on the axis of rotation. To simplify the discussion, whenever we refer to the inertia, the angular momentum or the angular velocity, we refer to their components on the axis of rotation. The controller takes as input the

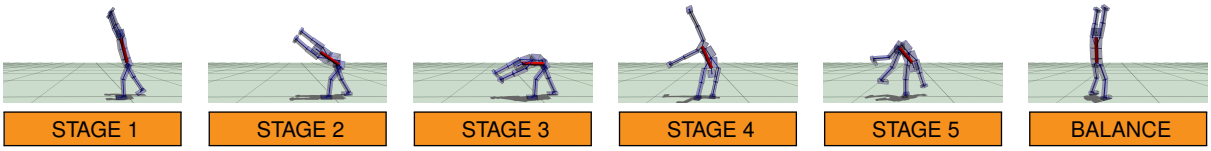

Figure 2: *The first frame of the five stages of the cartwheel controller and the last frame are shown. The red line connects the pelvis and the neck. The angle of this line is used to switch between stages.*

desired apex of the flip (α) and the desired inertia in the airborne stage (I_d), which is used to control the style of the flip. The controller is divided in two main stages: pre-airborne and airborne.

5.2.1. Pre-airborne stage

At first, an E_{COM} objective is used to place the character in a crouch position that is slowly tilting on its back. The feet are maintained to the ground with the $E_{rfootContact}$ and $E_{lfootContact}$ objectives. These contact objectives are deactivated when the character's torso is oriented towards α. The character is then directed to move towards the desired apex with $\bar{\mathbf{y}}_{COM} = \alpha$.

At this point, an E_{AM} objective is used to generate the necessary angular momentum for the character to flip. The character cannot land properly with insufficient or excessive angular momentum. Our approach to compute the angular momentum target is as follows. We begin by estimating the duration of the airborne stage of the flip: $t_{air} = 2\dot{\varepsilon}/\mathbf{g}$, where $\dot{\varepsilon}$ is the estimated velocity of the COM in the vertical axis at the start of the airborne stage and $\mathbf{g}$ is the gravitational constant. We set $\dot{\varepsilon}$ so that the height of the desired apex is achieved, namely, $\dot{\varepsilon} = \sqrt{2\mathbf{g}(\alpha^y - \varepsilon)}$, where ε depends on the character's stature. Assume that the flip is performed in the clockwise direction. It follows that the character's average angular velocity in the airborne stage is $w = \mu(\theta_{init}, \theta_{land})/t_{air}$, where θ_{init} is the estimated value of θ at the start of the airborne stage, θ_{land} is the desired value of θ at landing, and $\mu(\theta_{init}, \theta_{land})$ measures the clockwise angular distance between θ_{init} and θ_{land}. We can now specify the target for E_{AM}: $\bar{y}_{AM} = I_d w$. As can be seen in the accompanying video, changing I_d creates more or less tucked flips.

5.2.2. Airborne stage

The goal of the airborne controller is for the character to land with a desired orientation (θ_{land}) and inertia (I_{land}). The controller relies on the fact that θ is strictly monotonic in the airborne stage to position the character in the revolution.

The desired average angular velocity is given by $\bar{w}_{avg} = \mu(\theta, \theta_{land})/t_r$, where t_r is the estimated time remaining in the airborne stage. t_r can be easily calculated assuming that the character is a projectile $(\mathbf{c}, \dot{\mathbf{c}})$. Note that the controller remains time-invariant because t_r is calculated at each timestep from the character's state. The current angular velocity and the desired angular velocity at landing are given, respectively, by $w = |M|/I$ and $w_{land} = |M|/I_{land}$, where M denotes the angular momentum and $|\cdot|$ the absolute value. Let w_m denote angular velocity at time $t_r/2$ in the future. We approximate the character's average angular velocity with the composite trapezoid rule [CK12]:

$$w_{avg} = \frac{w + 2w_m + w_{land}}{4}. \tag{15}$$

The value of w_m is determined given that we want $w_{avg} = \bar{w}_{avg}$ to ensure a proper landing.

If $w < w_m$, the character is rotating too slowly, so we decrease the character's inertia to increase the angular velocity. If $w > w_m$, the character is rotating too quickly, so we increase the character's inertia to decrease the angular velocity. The inertia is modified with the rest pose of the joints in the set $\mathcal{L} = \{\text{elbow, knee, lumbar}\}$. We use the following control law:

$$\bar{r}_i = r_i + k_i(I - I_m), \tag{16}$$

where r_i and $\bar{r}_i$ denote the current and desired pose in the sagittal plane of joint $i \in \mathcal{L}$, k_i is a gain, and $I_m = |M|/w_m$. We constrain $\bar{r}_i$ to be within joint limits. When the character has completed most of its revolution, its feet are placed to the ground with (14). The balance controller is then used.

5.2.3. Discussion

In the accompanying video, we show that the flip controller succeeds for a wide variety of desired apex positions. Our simple balance controller is not suitable for the more aggressive flips, where the character lands with so much momentum that it needs to take a step to maintain balance. We also show that the controller is robust to external pushes of 500N for a duration of 0.1s to 0.3s. For example, if the character is pushed in the direction of the rotation while being airborne, it increases its inertia to slow down the rotation and successfully finish the movement. As the magnitude of the disturbances increases, the character can no longer sufficiently change its inertia to re-phase itself. We also show that the identical controller can be applied to characters with very different body proportions.

5.3. Diving

The goal of the diving controller is to have the character enter the water with a straight posture and with its arms raised

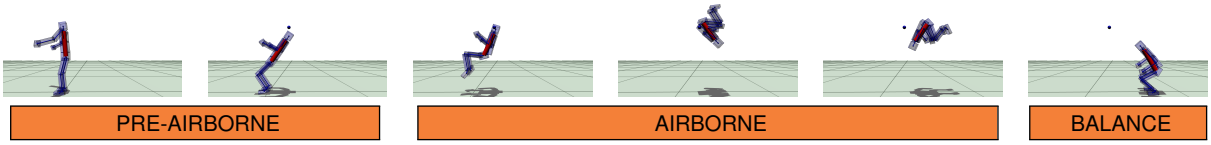

Figure 3: *In the pre-airborne stage, the backflip controller generates an appropriate amount of angular momentum for the desired apex and style of the flip. In the airborne stage, the controller modifies the character's inertia based on its current state and the desired landing position.*

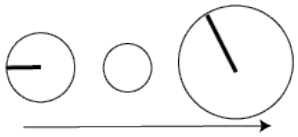

Figure 4: ***The airborne controller.*** *We model the character as a spinning disk with a radius that is proportional to the inertia. The line represents the orientation of the character. The leftmost disk models the current state of the character. The rightmost disk models the desired inertia and orientation at landing. The time between two consecutive disks is $t_r/2$, where t_r is the estimated time until landing. The airborne controller finds a target inertia (the middle disk) to connect the leftmost and rightmost disks.*

upwards. The controller takes as input the desired number of complete revolutions that the diver must accomplish before entry.

The diving controller is almost identical to the flip controller (Sec. 5.2). Here, we highlight the few differences. One, the values of θ_{land} and I_{land} are different because of the different desired landing positions. Second, we must take into account the number of complete revolutions to be performed when calculating the desired average angular velocity $\bar{w}_{avg}$. Lastly, the shoulders are now included in the set of joints $\mathcal{L}$, so that the character can have its arms raised upwards at entry.

In the accompanying video, we show that the diving controller succeeds for flips at different heights, angular momentum values, and desired number of complete revolutions. We show that the identical airborne controller can generate forward, backward, straight and armstand dives. Dives with straight knees are generated by removing the knee joint from $\mathcal{L}$. Twisting dives are generated with an E_{pose} objective to throw one arm up and one arm down in the coronal plane [Fro79, WH96]. The character spreads its arms to stop the twisting, and the diving controller as described in the above paragraph is used to finish the motion. The twisting dive controller succeeds for different heights, but there is much room for improvement. For instance, we did not investigate how the character should begin and stop twisting based on its angular momentum and inertia, which would make the controller more general. The multiple axes of rotation make the problem harder to analyze.

5.4. Backhandspring

The backhandspring is an acrobatic movement where a character performs a backwards jump, lands on its hands, and then gets back on its feet. The controller takes as input the desired apex of the motion. The pre-airborne stages of the backhandspring and backflip controllers (Sec. 5.2.1) differ only in the values of θ_{land} and I_d. Unlike backflips and dives, people do not significantly change their inertia in the airborne stage of backhandsprings. Hence, only the rest pose of the shoulders vary when the character is airborne in our controller. Specifically, the character raises its arms as a function of θ in order to land on its hands. The character enters a post-airborne stage when its hands are close enough to the ground and the hand contact objectives are activated. An E_{AM} objective is then used for the character to revolve around its hands. The character is brought back on its feet with the E_{foot} objectives presented in (14), before ending with the balance controller.

5.5. Pirouette

The pirouette is a famous ballet movement, where the body whirls rapidly about one leg. A foot target objective is used to maintain the toe of the supporting leg on the ground throughout the rotation. We bring the COM above the supporting toe with an E_{COM} objective. When the COM gets close enough to its desired position, an E_{AM} objective is used to generate angular momentum in the vertical axis. The number of revolutions that the character can achieve depends on the momentum generated. The rest pose is modified in order to bring the arms in a curved position and to bent the raised leg. When the COM is too far away from its desired position, the character no longer has the necessary momentum to continue whirling, and the balance controller is used. Note that we did not use a phase variable θ to parametrize the movement since the features are invariant with respect to the rotation.

5.6. Front Aerial

The front aerial controller can be divided in the pre-airborne, airborne and post-airborne stages (see Fig. 8). The pre-

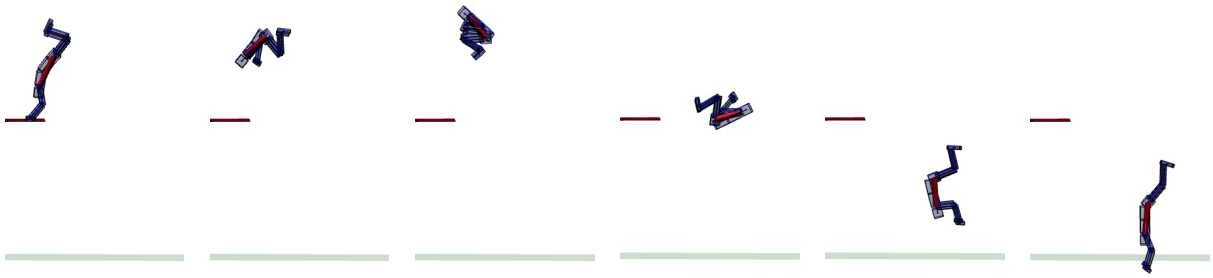

Figure 5: *The diving and flip controllers are almost identical, which demonstrates the generality of the control laws. We show that the identical control strategy succeeds for a wide variety of heights, flips, and styles.*

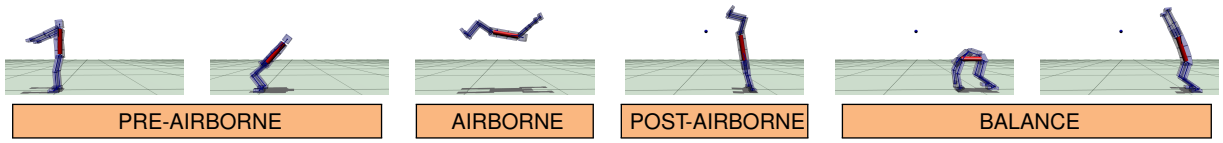

Figure 6: *The pre-aiborne stage of the backhandspring controller precisely regulates the angular momentum of the motion. In the airborne stage, the character raises its arms as the revolution progresses in order to ensure proper landing. In the post-airborne stage, the controller uses E_{AM} and E_{foot} to bring the character back on its feet.*

airborne stage differs from the corresponding stage of the flip controller (Sec. 5.2.1) only in the values of the θ_{land} and I_d parameters. In the airborne stage, the E_{pose} objective is used to split the legs. As the character gets near the end of its revolution, its feet are placed to the ground with (14), except that the desired heights decrease as the revolution progresses, similarly to (12).

5.7. Motion sequences

In the accompanying video, we provide examples of motion sequences that can be generated with our individual controllers. We sequence controllers together such that the state of the character at the end of each controller should be close enough to the desired initial state of the subsequent controller. In one example, the character successfully performs a series of backflips until its momentum becomes excessively high. In other examples, the character performs different types of movements in succession, for instance a backflip followed by a backhandspring.

6. Implementation details

The simulator uses Featherstone's algorithm and the semi-implicit scheme of Guendelman et al. [GBF03] ($1e^{-3}$ s). Ground contact uses a frictional ($\mu = 1$) inelastic impulse-based model. The system is single threaded and runs at $50-100\%$ real-time on a 2.7 Ghz core CPU. Simulation and control both run at 1kHz. Height and weight for the correspond to a 50^{th} percentile North American male. Skeletal dimensions/link masses are taken from Winter [Win04]. Link inertias are calculated using uniform density shapes scaled to match skeletal dimensions.

7. Discussion

We presented feature-based control algorithms for a wide variety of rotational movements, including aerials, cartwheels, dives, and flips, that do not require any input motion or offline optimization. Most of these movements have not previously been generated by physics-based methods without prior data. Our controllers are general and robust, important properties that are often lacking in previous results.

We use the following strategy when designing a controller. Our goal in the first iteration is to synthesize a single trajectory of the movement, without concern on making the controller time-invariant. On the second iteration, we substitute time with a phase variable and verify that it increases the robustness of the controller. On the last iteration, we attempt to generalize the controller so that it can synthesize a manifold of trajectories, e.g., cartwheels with different speeds and styles, aerials with different apexes, etc. In our experience, the first iteration can require a significant amount of trial-and-error, the second iteration is straightforward, and the last iteration requires insight in order to translate invariant properties of the movement into control strategies.

Our results lack some aspects of natural human motion.

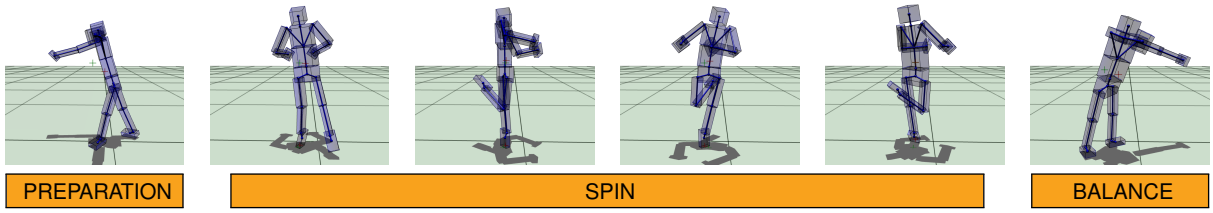

Figure 7: *The pirouette controller uses angular momentum to make the body whirl about the supporting leg.*

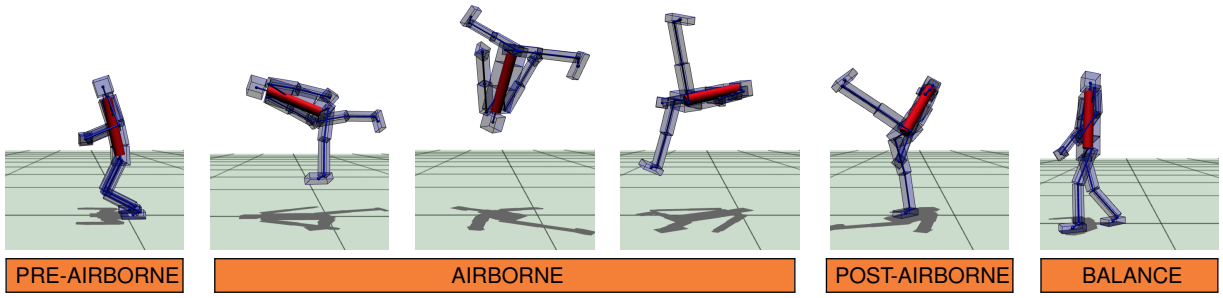

Figure 8: *The aerial controller begins like the flip controller. When the character is near the end of its revolution, the target vertical position of the feet decrease as the revolution progresses to ensure a proper landing.*

All models of motion control, including both synthetic controller models and those based on motion capture, will yield visual artifacts in some situations. As computational models of the human body become more comprehensive, however, the scope of realistic motions we will be able to simulate will grow. This will come at the cost of increased model dimensionality, so that reduced dimension models will become all the more important.

One important limitation of this work is that most of our rotations are planar. We believe that our phase parametrization can be extended to three-dimensional rotations by simply choosing θ to be an angle in one of the planes of rotation. However, designing controllers by hand becomes more difficult as the motion complexity increases. For instance, we were not able to synthesize some movements that are at the periphery of performance, such as certain ballet moves that require an enormous amount of precision. For these cases, developing alternative methods is essential. One approach could be to optimize the virtual constraints (7), but how to parametrize them is an open problem [GCAS10]. This could remove some of the artifacts in our synthesized motions, such as the character slightly tilting on the side in a flip or the cartwheel not being prefectly straight. Another approach could be to design a user interface to control the features, e.g., the trajectory of the hand or the COM. This could make control synthesis even more intuitive, closely resembling the way a spotter would work with an athlete.

The key element behind the robustness of the controllers is their time-invariance. Most of our movements were parametrized by a phase variable θ, which we chose to be the angle of the line between the neck and the pelvis. An open research question is how to automatically select θ in order to increase the controller's robustness. One difficulty comes from to the fact that the method of Poincaré sections, which is used to determine stability properties, is only applicable to periodic movements [GCAS10, Kha02].

This work is a decisive step towards closing the gap between the skill set of animated characters and the remarkable prowess of dancers and gymnasts. Our controllers could be used as building blocks for the synthesis of more complex rotational movements. Given a set of action-specific controllers, how can we combine arbitrary controllers together, i.e., not carefully chosen combinations as in Sec. 5.7? In other words, given controllers for flips, front aerials, and cartwheels, how can a character perform a flip, followed by a cartwheel, before finishing with a front aerial? One would need a method to augment and estimate the domain of attractions of our controllers, perhaps inspired by the work of Tedtrake et al. [Ted09] and Faloutsos et al. [FvdPT01].

Acknowledgments

We are grateful to Manfredi Maggiore for introducing us to virtual constraints and for very helpful discussions, and to the referees for their feedback.

References

[ABdLH13] Al Borno M., de Lasa M., Hertzmann A.: Trajectory optimization for full-body movements with complex

contacts. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013). 2

[AdSP07] ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. In *Proc. SCA* (2007). 1, 2, 3, 4

[Ame13] AMES A. D.: Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control* (2013). 2

[AP11] ABE Y., POPOVIC J.: Simulating 2D gaits with a phase-indexed tracking controller. *IEEE Computer Graphics and Applications 31*, 4 (2011). 2, 3

[ASvdP13] AGRAWAL S., SHEN S., VAN DE PANNE M.: Diverse motion variations for physics-based character animation. In *Proc. SCA* (2013). 2

[BMYZ13] BROWN D. F., MACCHIETTO A., YIN K., ZORDAN V.: Control of rotational dynamics for ground behaviors. In *Proc. SCA* (2013). 2, 3

[CK12] CHENEY E. E. W., KINCAID D. R.: *Numerical mathematics and computing*. Cengage Learning, 2012. 5

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-based locomotion controllers. *ACM Transactions on Graphics 29*, 4 (2010). 2, 3

[dSAP08] DA SILVA M., ABE Y., POPOVIĆ J.: Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27*, 3 (2008). 2, 3

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (2003). 2

[Fro79] FROHLICH C.: Do springboard divers violate angular momentum conservation? *American Journal of Physics 47* (1979). 6

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proc. SIGGRAPH* (2001). 8

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics 22*, 3 (2003). 7

[GCAS10] GRIZZLE J., CHEVALLEREAU C., AMES A., SINNET R.: 3D bipedal robotic walking: models, feedback control, and open problems. In *IFAC Symposium on Nonlinear Control Systems* (2010). 2, 3, 8

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. In *Proc. SIGGRAPH* (1997). 2

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *Proc. SIGGRAPH* (1995). 2

[HYL12] HA S., YE Y., LIU C. K.: Falling and landing motion control for character animation. *ACM Transactions on Graphics 31*, 6 (2012). 2

[JYL09] JAIN S., YE Y., LIU C. K.: Optimization-based interactive motion synthesis. *ACM Transactions on Graphics 28*, 1 (2009). 2

[Kha02] KHALIL H. K.: *Nonlinear systems*, vol. 3. Prentice hall Upper Saddle River, 2002. 8

[KKI06] KUDOH S., KOMURA T., IKEUCHI K.: Stepping motion for a human-like character to maintain balance against large perturbations. In *Proc. ICRA* (2006). 4

[Lie77] LIEGEOIS A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics 7*, 12 (1977). 2

[LKL10] LEE Y., KIM S., LEE J.: Data-driven biped control. *ACM Transactions on Graphics 29*, 4 (2010). 2

[LYvdP*10] LIU L., YIN K., VAN DE PANNE M., SHAO T., XU W.: Sampling-based contact-rich motion control. *ACM Transactions on Graphics 29*, 4 (2010). 2

[LYvdPG12] LIU L., YIN K., VAN DE PANNE M., GUO B.: Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Transactions on Graphics 31*, 6 (2012). 2

[MC13] MAGGIORE M., CONSOLINI L.: Virtual holonomic constraints for euler-lagrange systems. *IEEE Transactions on Automatic Control 58*, 4 (2013). 2

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics 29*, 4 (2010). 2

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3 (2009). 2

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C.: Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (2009), 80. 1, 2, 3

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics. 23*, 3 (2004). 2

[Ted09] TEDRAKE R.: LQR-Trees: feedback motion planning on sparse randomized trees. In *Proc. RSS* (2009). 8

[WFH09] WANG J., FLEET D., HERTZMANN A.: Optimizing walking controllers. *ACM Transactions on Graphics 28*, 5 (2009). 2

[WH96] WOOTEN W. L., HODGINS J. K.: Animation of human diving. *Computer Graphics Forum 15*, 1 (1996). 2, 6

[WHDK12] WANG J. M., HAMNER S. R., DELP S. L., KOLTUN V.: Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics 31*, 4 (2012). 2

[Win04] WINTER D.: *Biomechanics and motor control of human movement*, 3rd ed. Wiley, 2004. 7

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Proc. SIGGRAPH* (1988). 2

[Woo98] WOOTEN W.: *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology, 1998. 2

[WP09] WAMPLER K., POPOVIĆ Z.: Optimal gait and form for animal locomotion. *ACM Transactions on Graphics 28*, 3 (2009). 2

[WP10] WU J.-C., POPOVIĆ Z.: Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics 29*, 4 (2010). 2, 3

[WZ10] WU C.-C., ZORDAN V.: Goal-directed stepping with momentum control. In *Proc. SCA* (2010). 2, 4

[YL10] YE Y., LIU C. K.: Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics 29*, 3 (2010). 2

[YLv07] YIN K., LOKEN K., VAN DE PANNE M.: SIMBICON: simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (2007). 2