Falling and Landing Motion Control for Character Animation

Sehoon Ha Yuting Ye C. Karen Liu

Georgia Institute of Technology

Figure 1: *A simulated character lands on the roof of a car, leaps forward, dive-rolls on the sidewalk, and gets back on its feet, all in one continuous motion.*

Abstract

We introduce a new method to generate agile and natural human landing motions in real-time via physical simulation without using any mocap or pre-scripted sequences. We develop a general controller that allows the character to fall from a wide range of heights and initial speeds, continuously roll on the ground, and get back on its feet, without inducing large stress on joints at any moment. The character's motion is generated through a forward simulator and a control algorithm that consists of an airborne phase and a landing phase. During the airborne phase, the character optimizes its moment of inertia to meet the ideal relation between the landing velocity and the angle of attack, under the laws of conservation of momentum. The landing phase can be divided into three stages: impact, rolling, and getting-up. To reduce joint stress at landing, the character leverages contact forces to control linear momentum and angular momentum, resulting in a rolling motion which distributes impact over multiple body parts. We demonstrate that our control algorithm can be applied to a variety of initial conditions with different falling heights, orientations, and linear and angular velocities. Simulated results show that our algorithm can effectively create realistic action sequences comparable to real world footage of experienced freerunners.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Character Animation, Physics-based Animation, Optimal Control

Links: DL PDF

1 Introduction

One of the great challenges in computer animation is to physically simulate a virtual character performing highly dynamic motion with agility and grace. A wide variety of athletic movements, such as acrobatics or freerunning (parkour), involve frequent transitions between airborne and ground-contact phases. How to land properly to break a fall is therefore a fundamental skill athletes must acquire. A successful landing should minimize the risk of injury and disruption of momentum because the quality of performance largely depends on the athlete's ability to safely absorb the shock at landing, while maintaining readiness for the next action. To achieve a successful landing, the athlete must plan coordinated movements in the air, control contacting body parts at landing, and execute fluid follow-through motion. The basic building blocks of these motor skills can be widely used in other sports that involve controlled falling and rolling, such as diving, gymnastics, judo, or wrestling.

We introduce a new method to generate agile and natural human falling and landing motions in real-time via physical simulation without using motion capture data or pre-scripted animation (Figure 1). We develop a general controller that allows the character to fall from a wide range of heights and initial speeds, continuously roll on the ground, and get back on its feet, without inducing large stress on joints at any moment. Previous controllers for acrobat-like motions either precisely define the sequence of actions and contact states in a state-machine structure, or directly track a specific motion capture sequence. Both cases fall short of creating a generic controller capable of handling a wide variety of initial conditions, overcoming drastic perturbations in runtime, and exploiting unpredictable contacts.

Our method is inspired by three landing principles informally developed in freerunning community. First, reaching the ground with flexible arms or legs provides cushion time to dissipate energy over a longer time window rather than absorbing it instantly at impact. It also protects the important and fragile body parts, such as the head, the pelvis, and the tailbone. Second, it is advisable to distribute the landing impact over multiple body parts to reduce stress on any particular joint. Third, it is crucial to utilize the friction force generated by landing impact to steer the forward direction and control the angular momentum for rolling, a technique referred to as "blocking" in the freerunning community. These three principles outline the most commonly employed landing strategy in practice: landing with feet or hands as the first point of contact, gradually lowering the center of mass (COM) to absorb vertical impact, and

turning a fall into a roll on the ground, with the head tightly tucked at impact moment.

However, translating these principles to control algorithms in a physical simulation is very challenging. During airborne, the controller needs to plan and achieve the desired first point of contact and the angle of attack, in the absence of control over the characters global motion in the air. Instead of solving a large, nonconvex two-point boundary value problem, we develop a compact abstract model which can be simulated efficiently for real-time applications. To strike the balance between accuracy and efficiency, our algorithm replans the motion frequently to compensate the approximation due to the simplicity of the model. When the character reaches the ground, the controller needs to take a series of coordinated actions involving active changes of contact points over a large area of human body. Our algorithm executes three consecutive stages, impact, rolling, and getting-up by controlling poses, momentum, and contacts at key moments. Furthermore, the airborne and landing phases are interrelated and cannot be considered in isolation: the condition for a successful landing defines the control goals for the airborne phase while the actions taken during airborne directly impact the landing motion. We approach this problem in a reverse order of the action sequence: designing a robust landing controller, deriving a successful landing condition from this controller, and developing an airborne controller to achieve the landing condition.

We demonstrate that our control algorithm is general, efficient, and robust. We apply our algorithm to a variety of initial conditions with different falling heights, orientations, and linear and angular velocities. Because the motion is simulated in real-time, users can apply perturbation forces to alter the course of the character in the air. Our algorithm is able to efficiently update the plan for landing given the new situations. We also demonstrate different strategies to absorb impact, such as a dive roll, a forward roll, or tumbling. The same control algorithm can be applied to characters with very different body structures and mass distributions. We show that a character with unusual body shape can land and roll successfully. Finally, our experiments empirically showed that the algorithm induces smaller joint stress, except for the contacting end-effectors. In the worst case of our experiments, the average joint stress is still four times lower than landing as a passive ragdoll.

2 Related Work

Physical simulation of biped motion has many applications in computer animation and robotics. The most extensively studied movement is perhaps animal locomotion, which has many manifestations, including walking [Yin et al. 2007; da Silva et al. 2008; Muico et al. 2009; Lasa et al. 2010; Mordatch et al. 2010; Wu and Popović 2010; Coros et al. 2010; Coros et al. 2011], running [Yin et al. 2007; Mordatch et al. 2010; Wu and Popović 2010], jumping [da Silva et al. 2008; Lasa et al. 2010], swimming [Tan et al. 2011], crawling [Miller 1989], and flying [Wu and Popović 2003]. Although falling and landing motions are fundamentally different from continuous locomotion, our controller during the landing phase also needs to overcome the issue of balance and exploit contact forces to achieve control goals. In particular, our algorithm leverages the virtual force control which has been successfully applied to bipeds [Coros et al. 2010] and quadrupeds [Coros et al. 2011] locomotion.

In addition to locomotion, previous work has demonstrated that highly dynamic motions with a long ballistic phase can be synthesized using physics simulation or kinematic approaches. Hodgins *et al.* [1995; 1998] showed that carefully crafted control algorithms can simulate highly athletic motions, including diving, tumbling, vaulting, and leaping. Faloutsos *et al.* [2001] composed primitive controllers to simulate more complex motor skills, such as a kip-up move or a dive down stairs. Liu *et al.* [2010] successfully tracked contact-rich mocap sequences using a sampling-based approach. They showed that vigorous motions with complex contacts, such as a dive-roll or a kip-up move, can be dynamically simulated, provided full body mocap sequences as desired trajectories. Zhao and van de Panne [2005] provided a palette of parametrized actions to build a user interface for controlling highly dynamic animation. Other techniques directly edit ballistic motion sequences under the constraints imposed by conservation of momentum [Majkowska and Faloutsos 2007; Sok et al. 2010], or apply a hybrid method for synthesizing dynamic response to perturbation in the environment [Shapiro et al. 2003]. If the contact positions and timing are known, spacetime optimization techniques can also generate compelling dynamic motions [Liu and Popović 2002; Fang and Pollard 2003; Safonova et al. 2004; Sulejmanpašić and Popović 2004]. In this work, we take the approach of physical simulation, but we seek for a more general and robust control algorithm such that the controller can operate under a wide range of initial conditions and allow for runtime perturbations. Furthermore, our controller does not depend on any pre-scripted or captured motion trajectories.

Safe falling and landing for bipeds is a topic that receives broad attention in many disciplines. Robotic researchers are interested in safe falling from standing height for the purpose of reducing damages on robots due to accidental falls. Previous work has applied machine learning techniques to predict falling [Kalyanakrishnan and Goswami 2011], as well as using an abstract model to control a safe fall [Fujiwara et al. 2002; Fujiwara et al. 2007; Yun et al. 2009]. In contrast to the related work in robotics, our work focuses on falls from higher places. In those cases, control strategies during long airborne phase become critical for safe landing. We draw inspiration from kinesiology literature and sport practitioners. In particular, the techniques developed in freerunning and parkour community are of paramount importance for designing landing control algorithms capable of handling arbitrary scenarios [Edwardes 2009; HLJ 2011].

Many animals have astonishing capabilities to achieve different maneuvers in the air by manipulating their body articulations. Cats are known for landing with feet from any initial falling condition [Kane and Scher 1969; Montgomery 1993]. Lizards swing their tails to stabilize their bodies during a leap [Libby et al. 2012]. Pigeons reorient their bodies to achieve a sharp turn when flying at low speed [Ros et al. 2011]. These behaviors inspire scientists and engineers to develop intelligent devices and control algorithms. Our work has a similar goal that we study how human body can change shape in the air to reduce damage at landing.

3 Overview

We introduce a physics-based technique to simulate strategic falling and landing motions from a wide range of initial conditions. Our control algorithm reduces joint stress due to landing impact and allows the character to efficiently recover from the fall. The character's motion is generated through a forward simulator and a control algorithm that consists of an *airborne phase* and a *landing phase*. These two phases are related by an appropriate *landing strategy*, which describes the body parts used for the first contact with the ground, a desired landing pose, and an ideal landing condition that describes the relation between landing velocities and the angle of attack in successful landing motions. We develop two most common types of landing strategies: hands-first and feet-first, and introduce a sampling method to derive the ideal landing condition for each strategy.

At the beginning of a fall, the character first decides on a landing

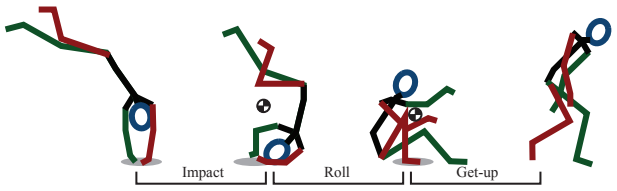


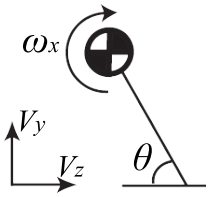
Figure 2: *Three stages in the landing phase.*

strategy. During the airborne phase, the character optimizes its moment of inertia to achieve the ideal landing condition. The landing phase is divided into three stages: impact, rolling, and getting-up (Figure 2). The impact stage begins when the character reaches the ground. During the impact stage, the character leverages the friction forces from the ground to control linear and angular momentum. After the COM moves beyond the hand contact area, the character switches to the rolling stage in which continuous change of contact carries out. In preparation for standing up, the character needs to maintain the rolling direction and plant its feet on the ground. When the COM passes through the first foot, the character starts to elevate the COM in order to compete the landing process in an upright position.

4 Landing Strategy

Given an initial condition at the beginning of a fall, the character can choose to land with the hands-first strategy or the feet-first strategy. In general, the hands-first strategy is chosen only for aesthetics purpose because it is less robust and suitable only for falls with planar angular momentum (about the pitch axis). In contrast, the feet-first strategy can handle a wide range of arbitrary initial conditions because it includes an extensive foot-ground contact duration to modulate the momentum before rolling. A landing strategy also includes a desired landing pose. Our algorithm only requires a partial pose to stretch the arms or legs at landing, depending on whether the hands-first or the feet-first strategy is chosen. We manually specify this partial pose for each strategy (Figure 3).

Figure 3: *The left and middle are the desired landing poses for the hands-first strategy and the feet-first strategy, respectively. The right is the ready-to-roll pose for the feet-first strategy, which we track only the upper body.*



An integral part of our landing strategy is the landing condition, a simple equation that compactly characterizes successful landing motions. If the character manages to turn a fall into a roll and gets back on its feet at the end of the roll, we consider it successful. Because a successful landing highly depends on whether the character is able to control the momentum at the moment of the first contact (T), our algorithm defines the landing condition as a relation between the global linear velocity $\mathbf{v}^{(T)}$, global angular velocity $\omega^{(T)}$, and the angle of attack $\theta^{(T)}$, which approximates the global orientation of the character. The actual coefficients of the landing condition depend on the design of the landing controller, which cannot be derived analytically, but can be learned from examples generated by the landing controller. We apply a sampling method, similar in spirit to the approach Coros *et al.* [Coros et al. 2009] presented for biped locomotion, to determine the landing condition for a particular landing strategy.

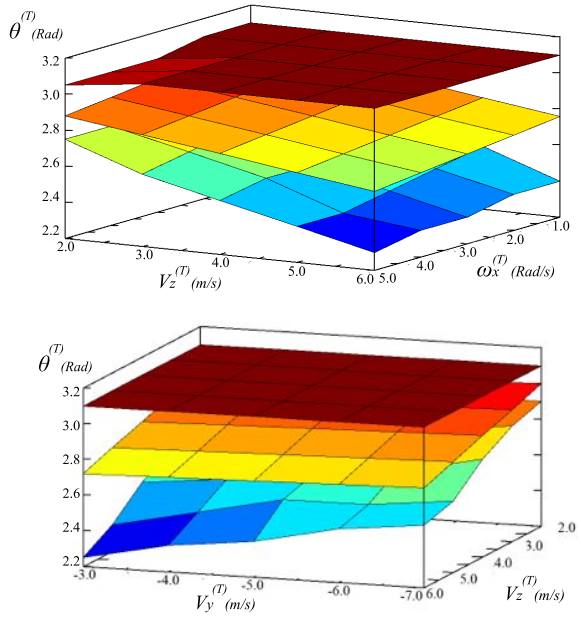


Figure 4: *Samples for hands-first landing strategy. Successful samples are bounded between top and bottom planes along $\theta^{(T)}$ axis. The middle plane, average of the two, indicates the linear relation of the ideal landing condition.*

For the hands-first strategy with planar motion, we consider a four-dimensional space spanned by $\theta^{(T)}$, $v_y^{(T)}$, $v_z^{(T)}$ and $\omega_x^{(T)}$. Given a sample in the parameter space, we run our landing controller to test whether the character can successfully get up at the end. Empirical results from thousands of random samples show that the successful region is mostly continuous and linear (Figure 4). We can bound the successful samples in the $\theta^{(T)}$ axis using two hyperplanes. Taking the average of the maximum and the minimum planes, we derive a linear relation between the angle of attack and the landing velocities as

$$\theta^{(T)} = a\, v_y^{(T)} + b\, v_z^{(T)} + c\, \omega_x^{(T)} + d \qquad (1)$$

where a, b, c, and d are the coefficients of the fitted hyperplane. Note that Equation (1) is a sufficient but not necessary condition for successful landing. Most points between the maximal and minimal hyperplanes also lead to successful landing motions. This means that even when the character cannot meet the landing condition exactly, it still has a good chance to land successfully. For the feet-first strategy, in theory, we need to consider all six dimensions of linear velocity and angular velocity. However, our empirical results show that non-planar velocities do not affect $\theta^{(T)}$ as long as they stay within a reasonable bound (Figure 5). As a result, the feet-first strategy is able to handle non-planner falling motion using the same parameters (but different coefficients) in Equation (1).

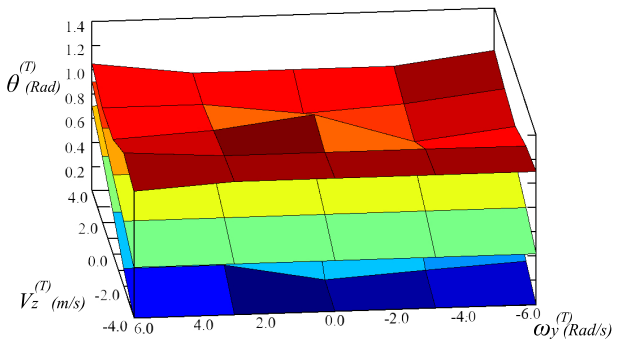


Figure 5: *Samples in the space of $v_z^{(T)}$, $\omega_y^{(T)}$, and $\theta^{(T)}$. The spinning velocity $\omega_y^{(T)}$ has minimal effect on the success of a sample.*

5 Airborne Phase

Once the character decides on a landing strategy, the goal of the airborne phase is to achieve the corresponding landing pose and landing condition. Because momentum is conserved in air, the linear velocity, the total airborne time T, as well as the angular momentum are already determined by the initial condition of the fall. However, the character can still control the angular velocity $\omega_x^{(T)}$ and the angle of attack $\theta^{(T)}$ by varying its pose (i.e. actuated degrees of freedom (DOFs) excluding the global position and orientation) to change the moment of inertia. To most effectively achieve the desired landing condition, we design our airborne algorithm based on the strategy employed in platform diving competition, where a highly trained athlete performs a sequence of predefined poses to manipulate the final orientation and angular velocity.

To this end, our airborne controller uses a PD servo to track a sequence of poses that lead to the ideal landing condition. The sequence of poses is replanned frequently to correct the errors caused by perturbation and numerical approximation. Each time the algorithm makes a new plan, an optimal sequence of poses from the current moment to the landing moment is computed. This sequence starts with the current pose $\mathbf{q}_0$ and ends at the desired landing pose $\mathbf{q}_T$ (determined by the landing strategy), with a duration of T seconds. Our control algorithm searches for an intermediate pose $\mathbf{q}^*$ and a duration Δt^*, such that the character can reach the ideal landing condition by changing to $\mathbf{q}^*$ immediately and holding the pose $\mathbf{q}^*$ for Δt^* seconds before changing to the final pose $\mathbf{q}_T$.

We formulate an optimization to solve for an intermediate pose $\mathbf{q}$ and its holding duration Δt that can best achieve the ideal landing condition. The cost function $g(\mathbf{q},\Delta t)$ is defined in Equation 2.

$$g(\mathbf{q},\Delta t) = \theta^{(T)}(\mathbf{q},\Delta t) - a\, v_y^{(T)} - b\, v_z^{(T)} - c\, \omega_x^{(T)}(\theta^{(T)}) - d \quad (2)$$

Note that $\omega_x^{(T)}$ is a function of $\theta^{(T)}$ because we need global orientation of the character at time T to compute the global angular velocity. If we can compute $\theta^{(T)}$, Equation (2) can be readily evaluated. Unfortunately, for a complex 3D multibody system, an analytical solution for $\theta^{(T)}$ is not available. We could resort to numerical simulation of the entire airborne phase, in which the character goes through $\mathbf{q}_0$, $\mathbf{q}^*$, and $\mathbf{q}_T$ subsequently. However, involving forward simulation of a full skeleton in the cost function is too costly for our real-time application. Instead, we simulate a simple proxy model with only six DOFs. When the character is holding a pose, the proxy model behaves like a rigid body with a fixed inertia. When the character transitions from one pose to another, we assume the inertia of the proxy model changes linearly within a fixed duration Δt_C ($\Delta t_C = 0.1s$ in our implementation). By simulating the proxy model for the duration of T, we obtain the angle of attack $\theta^{(T)}$ and angular velocity $\omega^{(T)}$ as follows.

$$\begin{aligned}\mathbf{R}(\theta^{(T)}) = \mathbf{R}(\theta^{(0)}) &+ \int_{t=0}^{\Delta t_c} [\mathbf{I}_A^{-1}(t)\mathbf{L}]\mathbf{R}(\theta^{(t)})dt \\ &+ \int_{t=\Delta t_c}^{\Delta t_c+\Delta t} [\mathbf{I}^{-1}(\mathbf{q},\theta^{(t)})\mathbf{L}]\mathbf{R}(\theta^{(t)})dt \\ &+ \int_{t=\Delta t_c+\Delta t}^{2\Delta t_c+\Delta t} [\mathbf{I}_B^{-1}(t)\mathbf{L}]\mathbf{R}(\theta^{(t)})dt \\ &+ \int_{t=2\Delta t_c+\Delta t}^{T} [\mathbf{I}^{-1}(\mathbf{q}_T,\theta^{(t)})\mathbf{L}]\mathbf{R}(\theta^{(t)})dt; \quad (3)\end{aligned}$$

$$\omega^{(T)} = \mathbf{I}^{-1}(\mathbf{q}_T,\theta^{(T)})\mathbf{L} \quad (4)$$

where $\mathbf{R}$ is the rotation matrix, $\mathbf{I}(\mathbf{q})$ is an inertia matrix evaluated at pose $\mathbf{q}$, and $\mathbf{L}$ is the angular momentum. $\mathbf{I}_A(t)$ is an interpolated inertia matrix between $\mathbf{I}(\mathbf{q}_0)$ and $\mathbf{I}(\mathbf{q})$, and similarly, $\mathbf{I}_B(t)$ is an interpolated matrix between $\mathbf{I}(\mathbf{q})$ and $\mathbf{I}(\mathbf{q}_T)$. The operator $[\,]$ represents the skew symmetric matrix form of a vector.

To formulate an efficient optimization for real-time application, we represent the domain of intermediate pose as a finite set of candidate poses, instead of a continuous high-dimensional Euclidean space. This simplification is justified because a handful of poses is sufficient to effectively change the moment of inertia of the character. As a preprocess step, our algorithm automatically selects the candidate set $\mathbf{Q}$ from a motion capture sequence in which the subject performs range-of-motion exercise. The selection procedure begins with a seed pose $\bar{\mathbf{q}}_0$ and increments the set by adding a new pose $\bar{\mathbf{q}}_{new}$ which maximizes the diversity of inertia (Equation 5). In our experiment, 16 poses are sufficient to present a variety of moment of inertia (Figure 6).

$$\bar{\mathbf{q}}_{new} = \underset{\mathbf{q}\in M}{\operatorname{argmax}}\{\min_{\bar{\mathbf{q}}_j\in Q} \|I(\mathbf{q}) - I(\bar{\mathbf{q}}_j)\|)\} \quad (5)$$

where M contains the poses in the range-of-motion sequence, Q contains the currently selected candidate poses, and $I(\mathbf{q})$ computes the inertia of pose $\mathbf{q}$.

To find optimal $\mathbf{q}^*$ and Δt^* for each plan, we start from the current pose as $\mathbf{q}_0$ and loop over each candidate pose in Q. For each candidate pose $\bar{\mathbf{q}}_i$, we search for the best Δt such that $g(\bar{\mathbf{q}}_i,\Delta t)$ is minimized. The search can be done efficiently using one-dimensional Fibonacci algorithm and the proxy-model simulation. The optimal intermediate pose $\mathbf{q}^*$ and its optimal duration Δt^* are used for airborne control.

By design, our algorithm trades off accuracy for efficiency; we use a fast but less accurate proxy-model simulation and a small set of predefined poses. Our algorithm is very efficient so that the character can frequently reassess the situation and replan new poses to correct any errors or adapt to unexpected perturbations.

The frequency of replanning can be determined differently for $\mathbf{q}^*$ and Δt^*. In our implementation, we replan $\mathbf{q}^*$ at a much lower frequency than Δt^* to avoid unnatural frequent change of poses. In addition, we stop replanning when the character is within 0.3 seconds away from the ground.

6 Landing Phase

During landing, the character braces for impact, executes rolling action, and gets up on its feet. Although these three stages take

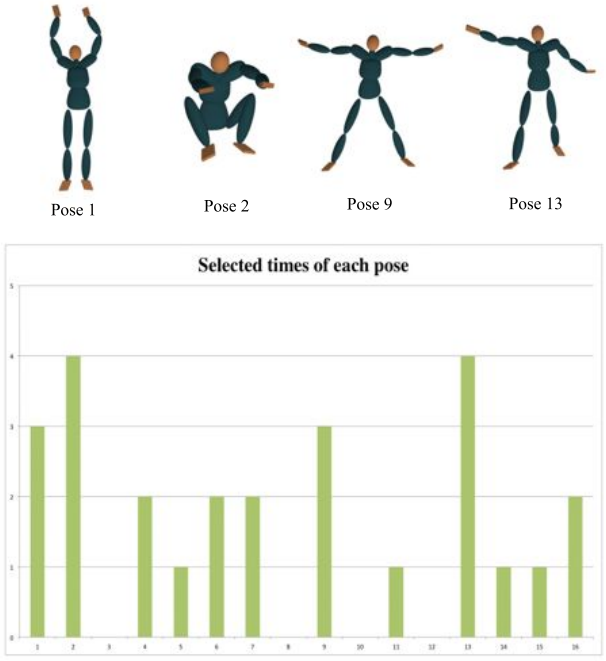


Figure 6: *Among 16 poses in* $\mathbf{Q}$*, pose 1, 2, 9, and 13 are frequently selected by the airborne controller*

very different actions, they share common control goals: modulating the COM and posing important joints. We apply the same control mechanism via *virtual forces* and *PID joint-tracking* to produce the final control forces for the forward simulator (Figure 7).

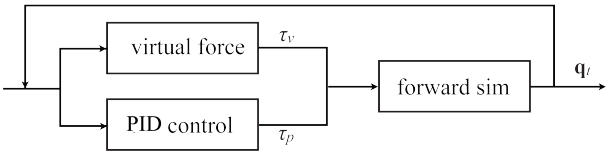


Figure 7: *Landing phase controller.*

Virtual forces are effective in controlling the motion of the COM. To achieve a desired acceleration of the COM, $\ddot{\mathbf{c}}$, we compute the virtual force as $\mathbf{f}_v = m\ddot{\mathbf{c}}$ where m is the mass of the character. The equivalent joint torque as if applying $\mathbf{f}_v$ to a point $\mathbf{p}$ on the body is $\tau_v = \mathbf{J}^T(\mathbf{p})\mathbf{f}_v$, where $\mathbf{J}(\mathbf{p})$ is the Jacobian computed at the body point $\mathbf{p}$. If $\mathbf{p}$ is on a body node in contact with the ground, we apply the opposite force ($\mathbf{f}_v = -m\ddot{\mathbf{c}}$) in order to generate a ground reaction force that pushes the COM in the desired direction. To prevent the character from using excessively large joint torques, we limit the magnitude of the sum of virtual forces. A successful landing motion also requires posing a few important joints at each of the three stages. We track these partial poses with PID servos: $\tau_p = k_p(\bar{q} - q) + k_i \int(\bar{q}_t - q_t)dt - k_v\dot{q}$, where k_p, k_i and k_v are the proportional, integral, and derivative gains respectively, and $\bar{q}$ is the desired joint angle. The final control torque is $\tau_v + \tau_p$. We limit the magnitude of the virtual force to $3000N$ to prevent excessive usage of joint torques.

6.1 Impact Stage

Impact stage is the most critical stage during landing, which requires careful control and execution. Human athletes tend to act like a spring to absorb the effect of impact by flexing their joints between the points of first contact and the COM. Meanwhile, they also utilize friction force from the ground contact to adjust forward linear momentum and angular momentum. Applying these principles, our algorithm utilizes virtual force technique to achieve contact forces for desired momentum. In addition, we use joint tracking to provide sufficient stiffness at contacting limbs and smooth transition to the next stage. If the character chooses the hands-first strategy, the final pose at the end of compression can seamlessly connect to the rolling stage. With the feet-first strategy, an additional "thrusting" step is required to transition to the rolling stage. We define a "ready-to-roll" pose that guides the character toward a rolling motion (Figure 3, Right). During this additional step, the character tracks the ready-to-roll pose while using its feet to thrust forward after its COM compressed to the lowest point (Figure 8).

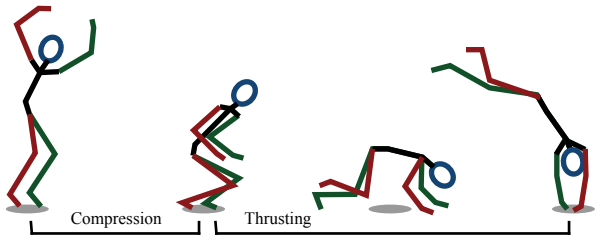


Figure 8: *Two-step impact stage for the feet-first strategy.*

Virtual force. The most important goal during the impact stage is to stop the downward momentum before the character tragically crashes into the ground. We do so by applying virtual forces to control the vertical position and velocity of the COM. In addition, our algorithm favors virtual forces that result in temporally smooth ground reaction forces to distribute the impact evenly over time. With these control goals, our algorithm aims to use constant acceleration of the COM to achieve the desired COM position $\bar{c}_y$ and velocity $\bar{\dot{c}}_y$ from the current state (c_y and $\dot{c}_y$).

$$\ddot{c}_y = \frac{1}{2}(\bar{\dot{c}}_y^2 - \dot{c}_y^2)/(\bar{c}_y - c_y) \tag{6}$$

A virtual force of $-m\ddot{c}_y$ in the vertical direction is then evenly distributed to the end-effectors that are in contact with the ground.

Virtual forces in the horizontal direction are important to achieve the desired forward linear momentum and angular momentum at the end of compression, or to achieve the desired forward thrust for the feet-first strategy. We use a simple feedback mechanism to compute the desired horizontal acceleration of the COM.

$$\ddot{c}_{x/z} = k_v(\bar{\dot{c}}_{x/z} - \dot{c}_{x/z}) \tag{7}$$

where $\bar{\dot{c}}_{x/z}$ is the desired COM velocity in forward and lateral directions and k_v is the damping coefficient. The corresponding virtual force is distributed to the contacting end-effectors inversely proportional to their distances to the COM.

Joint tracking. In addition to virtual forces, we use PID servos to maintain joint angles of the torso and limbs that are not in contact, while limbs in contact with the ground act like viscous dampers (PID control with a zero spring coefficient). We also use PID control to keep the chin tucked to reduce the chance of the head impacting the ground. Please see Table 1 for all the parameters in our implementation. We set the constant integral gain k_i of contacting limbs as 50, and 0 for all other joints.

Table 1: *Control parameters.*

	Hip	Lower spine	Upper spine	Neck	Knee
k_p	90.0	300.0	180.0	10.0	60.0
k_d	20.0	60.0	40.0	2.0	13.0
	Ankle	Clavicle	Shoulder	Elbow	Wrist
k_p	15.0	180.0	120.0	60.0	9.0
k_d	6.0	40.0	27.0	13.5	4.0
$\bar{c}_y$	$\dot{\bar{c}}_y$	$\dot{\bar{c}}_{x/z}$	k_v	k_p (Eq 8)	ω_{MAX}
0.4m	0.0m/s	4.0m/s	500	800	3.3 Rad/s

6.2 Rolling Stage

Once the character's COM passes the hand-ground contact area with sufficient forward linear and angular momentum, rolling becomes a relatively easy task. As long as the character is holding a pose with a flexed torso, a reasonable rolling motion will readily carry out. If the character wishes to land back on its feet and get up after rolling, it must also maintain forward momentum and lateral balance during the roll.

Virtual force. To this end, we apply a virtual force to guide the horizontal position of the COM toward the feet area, while restricting it above the support polygon formed by contact points. The virtual force is applied on the character's hands so that it can use the entire upper body to maintain momentum and balance. The virtual force produces the desired acceleration of the COM computed using a feedback mechanism:

$$\ddot{c}_{x/z} = k_p(\bar{c}_{x/z} - c_{x/z}) \tag{8}$$

where the desired position $\bar{c}$ is set to be the location of the left foot.

Joint tracking. During rolling, the character tracks a simple pose to tuck the head, flex the torso, and position the legs appropriately. We treat legs asymmetrically to both facilitate momentum control and improve the aesthetics of the motion. When the character rolls on its back, it brings the left knee closer to the chest and casually stretches the right leg. This arrangement helps the character to regulate the angular velocity using the right leg while getting ready to stand up on its left foot. Based on the forward angular velocity at the beginning of the rolling stage, we adjust the desired tracking angles for the right knee as:

$$\theta_R = max((1 - \omega_x/\omega_{MAX})\pi, 0) \tag{9}$$

6.3 Getting-Up Stage

The last stage of landing phase is to stand up using the remaining forward momentum. When the COM passes the foot contact, the character will start to elevate its COM to a desired height.

Virtual force. Similar to previous stages, we again apply virtual forces on the feet and the hands to control the vertical and the horizontal positions of the COM respectively. We compute $\ddot{c}_y$ using the same formula from Section 6.1 with different desired height of the COM. For $\ddot{c}_{x/z}$, we use the same formula as in Section 6.2.

Joint tracking. During the getting-up stage, our algorithm simply tracks the torso and the head to straighten the spine and untuck the chin.

7 Results

To evaluate the generality of our algorithm, we simulated landing motions with a wide range of initial conditions (Table 2), various landing styles (hands-first, feet-first, consecutive rolls), and different skeleton models. We also demonstrated that our algorithm is robust to unpredicted runtime perturbations and different physical properties of the landing surface. Please see the accompanying video to evaluate the quality of our results.

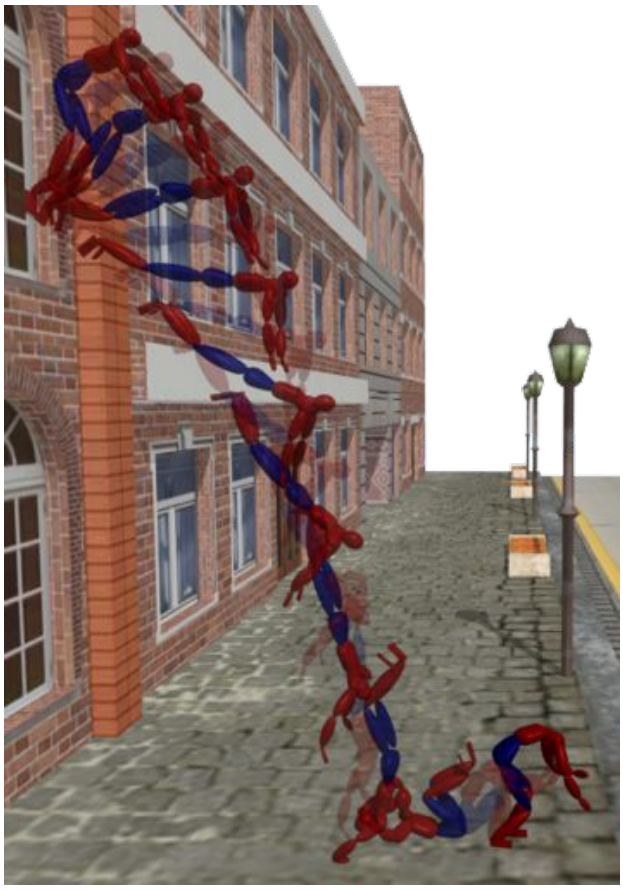


Figure 9: *Hands-first landing motion.*

Feet-first landing strategy. The most recommended landing strategy from freerunning community is the feet-first landing. Our results verify that the feet-first landing strategy is indeed very robust for falls with arbitrary linear and angular momentum. There are two key advantages of using feet as the first point of contact. First, average human has longer and stronger legs than arms. Using legs to land provides more time and strength to compress and absorb vertical impact. Second, the feet-first strategy has an additional thrusting step after compression and before rolling stage. During the thrusting step, the character can utilize the contact forces to drastically change the linear and angular velocity in preparation for rolling. Our results show that a successful forward roll can be carried out even when the character is falling with backward and lateral linear velocity or nonplanar angular velocity.

For the feet-first strategy, the coefficients of the landing condition in Equation (1) are: $a = -0.01, b = -0.06, c = -0.03$, and $d = 0.45$. When the character transitions to the rolling stage, we specified an asymmetric ready-to-roll pose to increase the visual appeal of the motion.

Hands-first landing strategy. Using hands as the first point of contact can generate visually pleasing stunts (Figure 9). For falls

Table 2: *Initial conditions of the examples shown in the video (in order of appearance)*

Hands-first landing strategy						
$\bar{C}_y$(m)	v_x(m/s)	v_y(m/s)	v_z(m/s)	ω_x(Rad/s)	ω_y(Rad/s)	ω_z(Rad/s)
10.6	0.0	0.0	4.0	8.7	0.0	0.0
5.8	0.0	0.0	2.3	5.0	0.0	0.0
10.6	0.0	0.0	6.0	2.5	0.0	0.0
2.5	0.0	0.4	8.0	5.0	0.0	0.0
Feet-first landing strategy						
$\bar{C}_y$(m)	v_x(m/s)	v_y(m/s)	v_z(m/s)	ω_x(Rad/s)	ω_y(Rad/s)	ω_z(Rad/s)
6.0	0.0	0.0	5.0	4.0	-1.0	-5.8
2.7	0.0	-1.0	0.0	0.0	0.0	0.0
5.5	1.0	0.0	0.0	0.0	5.0	0.0
9.6	-2.0	0.0	-3.5	0.9	2.1	-3.9

with dominant planar velocity (v_z and ω_x), the hands-first strategy performs as well as the feet-first strategy. However, when the initial condition has large lateral linear momentum or angular momentum in yaw and roll axes, the hands-first strategy becomes less robust. Unlike the feet-first strategy, which has an additional thrusting step, the hands-first strategy is unable to change forward direction drastically after landing. This imposes stringent conditions on the contact forces because, in order to roll successfully, the contact forces must counteract non-planner momentum, while stopping downward momentum and maintaining forward momentum. Such forces usually violate the unilateral constraint of ground reaction force.

For the hands-first strategy, the coefficients of the landing condition are: $a = -0.01, b = -0.06, c = -0.03$, and $d = 3.08$. Note that the coefficients are identical to those of the feet-first strategy except for the constant term, indicating that the gradient of the angle of attack with respect to the landing velocity is the same between feet-first and hands-first landing strategies.

Consecutive rolls. Once the character starts rolling, it is rather effortless to continue on. By looping the end of the rolling stage back to the beginning, we showed that the character was able to make two consecutive rolls to break a fall with large forward speed. Falling on multiple surfaces is also easy to simulate using our controller. One example demonstrated a continuous sequence of the character landing on the roof of a car, leaping forward, landing again on the sidewalk, and finishing with a dive roll (Figure 1). With our controller, a variety of impressive action sequences can be generated easily without any recorded or pre-scripted motions.

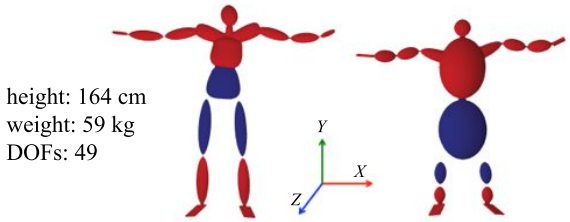


Figure 10: *Left: The character model used for most examples. Right: A character with a disproportionately large torso and short legs.*

Different skeleton models. The character model we used to generate most examples has a height of 164cm, a weight of 59 kg, and 49 DOFs. The controllers designed for this character can be applied to a drastically different character whose torso is twice as long and twice as wide, comparing to the default character. It also has very short legs and a small head (Figure 10). We tested both hands-first and feet-first landing strategies on this new character. The results are similar in quality to the default character, although the new character hits its head on the ground because it is difficult to tuck the head with such a short neck. All the control parameters remain the same for the second character, except for $\bar{c}_y$ increasing by $5cm$ and the desired landing angle increasing by $0.25rad$.

Runtime perturbations. One great advantage of physical simulation is that the outcome can be altered on the fly based on user interactions. We demonstrated the interactivity of our simulation in two different ways. First, the user can directly "drag" the character to a different location or orientation when the character is in the air. This example shows off robustness and efficiency of our airborne controller. As the character being relocated, it starts to recalculate and finds a new plan to execute in real-time. Second, we let the user shoot cannons at the character as a source of external forces. When a cannon hits the character, it exerts force and torque on the character, causing a passive response followed by active replanning and execution.

Different landing surfaces. We tested our controller on surfaces with different elasticities and friction coefficients. When the character lands on an elastic surface, such as a gymnastic floor or a trampoline, the character tumbles in the air instead of rolling on the ground. We generated a continuous sequence where the character stopped the fall on an elastic surface by tumbling three times and finishing with a forward roll. This example shows that various interesting acrobatic sequences can be generated by simply concatenating our falling and rolling controllers repeatedly. In another example, we reduced the friction coefficient to simulate an icy surface. The character was able to use the same control algorithm to roll, but failed to stand up at the end.

7.1 Evaluation

Performance. All the results shown in the video were produced on a single core of 3.20GHz CPU. Our program runs at 550 frames per second. The bottleneck of the computation is the optimization routine in the airborne controller. We use Open Dynamic Engine to simulate the character. The time step is set at 0.2 millisecond, and runs the airborne optimization in 50 Hz.

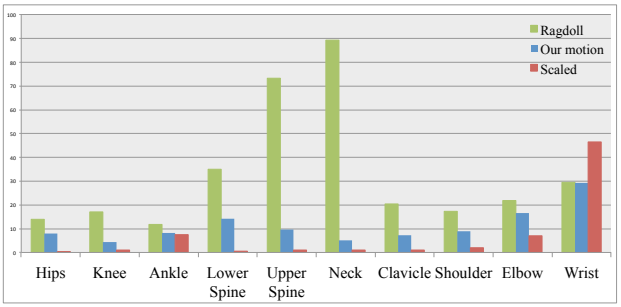


Figure 11: *Maximal stress for each joint from a hands-first landing motion. Results are quantitatively similar across all of our simulations. Green: Ragdoll motion. Blue: Our motion. Orange: Joint stress scaled by mass.*

Joint stress. We approximated joint stress as the constraint force that holds two rigid bodies together at a joint. For each joint, we

computed the maximal joint stress during the landing phase (Figure 11). We observed that, in most trials, the joints which endure the most impact are those connected to contacting end-effectors (i.e. hands or feet). The spine joints (lumbar and thoracic vertebrae) and hip joints are also subject to large impact. However, when we scaled each joint by the total mass it supports (e.g. the hip joint supports the mass of the entire leg), we found that the joint stress has low variance across the entire character's body, with the exception of the joints near the end-effectors.

When we compared the joint stress between our motion and a passive ragdoll motion with the same initial condition, the ragdoll motion caused much more damage on the neck and the spine (Figure 11). In fact, the only joints that endured similar amount of stress were those used for the first point of contact (e.g. wrists or ankles). These results validate that our controller indeed produces safer landing motion and protects important body parts. We repeated the experiments for different initial conditions. In the worst case of our experiments, the average joint stress is still four times lower than landing as a passive ragdoll. The data also show that our controller generates less damaging landing motion even when the character cannot roll successfully, such as dropping from 20 meters.

Comparison with video footages. We compared our simulated motion side-by-side with a collection of video footages ([APR 2011]). The simulations are based on the same landing strategy and our best guess of the initial conditions from the videos. Although it is not possible to achieve identical motions, results show that our motion is qualitatively similar to the video footages.

7.2 Limitations

The main limitation of our work is the lack of balance control after the character stands up. There are many existing balance control algorithms we could implement. However, we chose to defer the implementation until we decide on what the character's next action should be. In the freerunning scenario, the character transitions to running motion seamlessly right after a roll. If freerunning is our goal, we would modify the current get-up control algorithm to provide more forward thrust. Other possibilities of the next action include walking, stepping, jumping, or standing still. Different next actions will result in different balance strategies. Ideally, a character should be equipped with motor skills to execute all different balance strategies and autonomously determines which strategy to execute, but this is considered out of the scope of this work.

Another limitation is the predefined landing pose for each landing strategy. This inflexibility can negatively affect the character's ability to adapt to different environments. For example, if the character lands on a narrow wall, the landing pose needs to be adjusted on the fly. One possible solution is to use a simple inverse kinematics method to compute desired joint angles before landing.

8 Conclusion

We introduced a real-time physics-based technique to simulate strategic falling and landing motions. Our control algorithm reduces joint stress due to landing impact and allows the character to efficiently recover from the fall. Given an arbitrary initial position and velocity in the air, our control algorithm determines an appropriate landing strategy and an optimal sequence of actions to achieve the desired landing velocity and angle of attack. The character utilizes virtual forces and joint-tracking control mechanisms during the landing phase to successfully turn a fall into a roll. We demonstrated that our control algorithm is general, efficient, and robust by simulating motions from different initial conditions, characters with different body shapes, different physical environments, and scenarios with real-time user perturbations. The algorithm guides the character to land safely without introducing the large stress at every joint except for the contacting end-effectors.

Freerunning is a great exemplar to demonstrate human athletic skills. Those wonderfully simple yet creative movements provide a rich domain for future research directions. Based on the contribution of this work, we would like to explore other highly dynamic skills in freerunning, such as cat crawl, underbar, or turn vault. These motions are extremely interesting and challenging to simulate because they involve sophisticated planning and control in both cognitive and motor control levels, as well as complex interplay between the performer and the environment.

The landing strategies described in this work are suitable for highly dynamic activities, but not optimal for low-clearance falls from standing height. There is a vast body of research work in biomechanics and kinesiology studying fall mechanics of human from standing height. One future direction of interest is to integrate this domain knowledge with physical simulation tools to explore new methods for fall prevention and protection.

References

2011. *Advanced Parkour Roll Techniques, http://youtu.be/bbs7wDqViY4*.

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

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Trans. Graph. 29* (July), 130:1–130:9.

COROS, S., KARPATHY, A., JONES, B., REVERET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. 1–11.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. In *ACM SIGGRAPH 2008 papers*, 82:1–82:10.

EDWARDES, D. 2009. *The Parkour and Freerunning Handbook*. It Books, August.

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. on Graphics (SIGGRAPH)* (July), 417–426.

FUJIWARA, K., KANEHIRO, F., KAJITA, S., KANEKO, K., YOKOI, K., AND HIRUKAWA, H. 2002. UKEMI: Falling motion control to minimize damage to biped humanoid robot. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, IEEE, vol. 3, 2521–2526.

FUJIWARA, K., KAJITA, S., HARADA, K., KANEKO, K., MORISAWA, M., KANEHIRO, F., NAKAOKA, S., AND HIRUKAWA, H. 2007. An optimal planning of falling motions of a humanoid robot. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, no. Table I, 456–462.

2011. *How to Land a Jump in Parkour, http://www.wikihow.com/Land-a-Jump-in-Parkour*.

Figure 12: *Feet-first landing motion.*

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

KALYANAKRISHNAN, S., AND GOSWAMI, A. 2011. Learning to predict humanoid fall. *International Journal of Humanoid Robotics 8*, 2, 245–273.

KANE, T. R., AND SCHER, M. P. 1969. A dynamical explanation of the falling cat phenomenon. *Int J Solids structures*, 55, 663–670.

LASA, M. D., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics (TOG) 29*.

LIBBY, T., MOORE, T. Y., CHANG-SIU, E., LI, D., COHEN, D. J., JUSUFI, A., AND FULL, R. J. 2012. Tail-assisted pitch control in lizards, robots and dinosaurs. *Nature advance online publication* (January).

LIU, C. K., AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. on Graphics (SIGGRAPH) 21*, 3 (July), 408–416.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics (TOG) 29*, 4, 128.

MAJKOWSKA, A., AND FALOUTSOS, P. 2007. Flipping with physics: motion editing for acrobatics. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 35–44.

MILLER, G. 1989. Goal-directed snake motion over uneven terrain. In *Computer Graphics '89*, Blenheim Online Ltd., Middlesex HA5 2AE, UK, Blenheim Online Publications, 257–272.

MONTGOMERY, R. 1993. Gauge theory of the falling cat. In *Dynamics and Control of Mechanical Systems*, American Mathematical Society, M. J. Enos, Ed., 193–218.

MORDATCH, I., DE LASA, M., AND HERTZMANN, A. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Trans. Graph. 29* (July), 71:1–71:8.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. In *ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, SIGGRAPH '09, 81:1–81:9.

ROS, I. G., BASSMAN, L. C., BADGER, M. A., PIERSON, A. N., AND BIEWENER, A. A. 2011. Pigeons steer like helicopters and generate down- and upstroke lift during low speed turns. *Proceedings of the National Academy of Sciences (PNAS) 108*, 50.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensinal, behavior-specific spaces. *ACM Trans. on Graphics (SIGGRAPH) 23*, 3, 514–521.

SHAPIRO, A., PIGHIN, F., AND FALOUTSOS, P. 2003. Hybrid control for interactive character animation. *Computer Graphics and Applications*, 455–461.

SOK, K. W., YAMANE, K., LEE, J., AND HODGINS, J. 2010. Editing dynamic human motions via momentum and force. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

SULEJMANPAŠIĆ, A., AND POPOVIĆ, J. 2004. Adaptation of performed ballistic motion. *ACM Trans. on Graphics 24*, 1.

TAN, J., GU, Y., TURK, G., AND LIU, C. K. 2011. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, 58:1–58:12.

WOOTEN, W. L. 1998. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology.

WU, J., AND POPOVIĆ, Z. 2003. Realistic modeling of bird flight animations. vol. 22, 888–895.

WU, J.-C., AND POPOVIĆ, Z. 2010. Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph. 29* (July), 72:1–72:10.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. In *SIGGRAPH*, 105.

YUN, S.-K., GOSWAMI, A., AND SAKAGAMI, Y. 2009. Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. *2009 IEEE International Conference on Robotics and Automation* (May), 781–787.

ZHAO, P., AND VAN DE PANNE, M. 2005. User interfaces for interactive control of physics-based 3d characters. *I3D: ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*.