Practical Character Physics for Animators

Ari Shapiro ■ *Institute for Creative Technologies*

Sung-Hee Lee ■ *Gwangju Institute of Science and Technology*

This system lets animators improve unrealistic motions in 3D animation by visualizing motions' physical properties such as the center of mass, angular momentum, and zero-moment point, and by comparing the original created path to a generated physics-based path. Animators then modify the original path to match the generated path.

Realism is important in the production of live-action visual effects when animated characters occupy the same scene as the live actors and the live environment. In such scenarios, a virtual character's movements must visually match the behavior and movements in the live environment, or the discrepancy will be obvious to the viewer. For example, a character who is jumping and thus being brought to the ground by gravity must visually match an object that's being thrown in the same scene under the same gravitational force.

However, the traditional tools for creating 3D character animation don't include dynamical information, which means that the dynamics of character motion can exist only implicitly in the animation framework. Some animation systems incorporate physical simulation of rigid and nonrigid solids, fluids, gases, and characters.[1–4] In addition, technical animators often apply dynamic effects during postprocessing, such as creating the secondary-motion effects of muscle bulging and hair bouncing. However, the vast majority of character animators create most character movement through kinematic means. Character animation in feature films often requires a fine-grained level of control over all parts of the character's movement that can't be achieved by current character dynamics simulation methods. In addition, a particular shot's constraints might require that an animated character's motion violate the laws of physics. For example, this might occur when a character needs to move unnaturally to stay in a camera's view. Also, animators are typically trained using kinematic tools and thus develop a high level of proficiency using them.

So, conventional keyframe animation has remained the method of choice for animation studios. To generate realistic-looking motion, professional animators typically combine methods such as keyframing, inverse kinematics, and other traditional tools such as curve editors. Animators also frequently use reference motion, such as videos of people and creatures performing a motion they need to replicate. These references help animators approximate the dynamics of motion, because traditional rigs don't include dynamical aspects such as masses and moments of inertia. Thus, animators must create physical plausibility without the direct input of these physical aspects.

We've developed an interactive system that helps animators create more physically plausible character motions. To this end, it lets animators view a character's or object's motion as if it obeyed the laws of physics. Specifically, our system produces visualizations of dynamical properties, such as the center of mass, momentum, and balance. For example, it creates a physically accurate ballistic motion path alongside the original kinematic path. By comparing the two paths and viewing the additional dynamical information, animators can adjust the original animation to create a more physically correct animation. In addition, our system can automatically alter the animation to account for the discrepancy between the original animation and the physically correct animation. This lets nonanimators quickly correct existing animation without an animator's input.

Published by the IEEE Computer Society 0272-1716/11/$26.00 © 2011 IEEE

Related Work in Realistic Animation

The main article focuses on helping animators create physically realistic character animation by visualizing dynamical properties of keyframe animation. For details on the mathematical preliminaries, see *A Mathematical Introduction to Robotic Manipulation*.[1] In contrast, most physics-based animation techniques deal with automatic generation of animation with minimal user inputs. Space-time optimization approaches automatically create physically plausible animation by solving optimization problems subject to physical and other constraints.[2,3] Another approach develops algorithms that control a character under physical simulation. Researchers have constructed such dynamic controllers to let characters perform simple athletic maneuvers,[4] swimming,[5] stable walking cycles,[6] reactive motions such as falling,[7] and other motions such as breathing and grasping.[8,9] Such approaches can work in concert with kinematic animation[10] and motion capture.[11] Researchers have also used physical simulation to create realistic secondary motions.[12,13]

These physics-based techniques pursue a promising method of animation production, but the industry has yet to widely employ them, for the reasons we mention in the main article. Also, physics-based animation is often computationally heavy, which prevents its use in interactive authoring environments. In contrast, we developed our approach to improve conventional keyframing-based techniques' physical realism by visualizing kinematic animation's dynamic properties. So, the animation industry can readily employ our system.

Our approach visualizes physical properties such as the center of mass, momentum, and the center of pressure (COP). Researchers have explicitly used these properties to increase the physical realism of existing animation. Using motion capture data, Anna Majkowska and Petros Faloutsos created flip and backflip motions that obey momentum preservation laws.[14] Seyoon Tak and Hyeong-Seok Ko[15] and Hyun Joon Shin and his colleagues[16] enforced a zero moment point (ZMP) constraint for locomotion or a linear- or angular-momentum constraint for ballistic motions (COP and ZMP coincide for locomotion on the flat ground.[17]). Adnan Sulejmanpasić and Jovan Popović produced a physically valid ballistic motion by adapting an existing motion to new constraints.[18]

Despite these techniques' usefulness, many animators still prefer to manually edit motion because they need to control various aspects of it, ranging from kinematic constraints to the animation's overall style. Our approach differs from the ones we just described in that, again, we're interested primarily in helping animators create physically plausible animation and satisfy various other requirements, not in actually creating such motions.

Other research has shown that viewers can be sensitive to certain types of errors in ballistic motion.[19,20] Our user studies show that animators often generate animations that exceed these thresholds, thus creating perceptibly implausible motion.

References

1. R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.
2. A. Witkin and M. Kass, "Spacetime Constraints," *Proc. Siggraph*, ACM Press, 1988, pp. 159–168.
3. C.K. Liu, A. Hertzmann, and Z. Popović, "Learning Physics-Based Motion Style with Nonlinear Inverse Optimization," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 1071–1081.
4. J. Hodgins et al., "Animating Human Athletics," *Proc. Siggraph*, ACM Press, 1995, pp. 71–78.
5. P.-F. Yang, J. Laszlo, and K. Singh, "Layered Dynamic Control for Interactive Character Swimming," *Proc. 2004 ACM Siggraph/Eurographics Symp. Computer Animation*, Eurographics Assoc., 2004, pp. 39–47.
6. J. Laszlo, M. van de Panne, and E. Fiume, "Limit Cycle Control and Its Application to the Animation of Balancing and Walking," *Proc. Siggraph*, ACM Press, 1996, pp. 155–162.
7. P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable Controllers for Physics-Based Character Animation," *Proc. Siggraph*, ACM Press, 2001, pp. 251–260.
8. V.B. Zordan et al., "Breathe Easy: Model and Control of Simulated Respiration for Animation," *Proc. 2004 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA 04), Eurographics Assoc., 2004, pp. 29–37.
9. N.S. Pollard and V.B. Zordan, "Physically Based Grasping Control from Example," *Proc. 2005 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA 05), ACM Press, 2005, pp. 311–318.
10. A. Shapiro, F.H. Pighin, and P. Faloutsos, "Hybrid Control

The system's main purpose is not to generate physically correct motion automatically but to inform animators of the changes necessary to make motions physically correct. (For a look at some automatic systems, see the "Related Work" sidebar.) Because the animator ultimately has complete control over the extent to which the original animation is changed, the system is easily adaptable to a professional animator's toolset.

Improving Physical Realism

We've found that simply visualizing the physical properties helps animators create more realistic animation. In addition, we've integrated two tools into keyframe-based animation-authoring software. A ballistic-path tool lets animators easily create or modify ballistic animation. An angular-momentum tool rotates a character's global orientation to achieve the desired angular momentum.

for Interactive Character Animation," *Proc. 11th Pacific Conf. Computer Graphics and Applications*, IEEE CS Press, 2003, pp. 455–461.
11. V.B. Zordan et al., "Dynamic Response for Motion Capture Animation," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 697–701.
12. M. Neff and E. Fiume, "Modeling Tension and Relaxation for Computer Animation," *Proc. 2002 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA 02), ACM Press, 2002, pp. 81–88.
13. M. Neff and E. Fiume, "Methods for Exploring Expressive Stance," *Proc. 2004 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA 04), Eurographics Assoc., 2004, pp. 49–58.
14. A. Majkowska and P. Faloutsos, "Flipping with Physics: Motion Editing for Acrobatics," *Proc. 2007 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA '07), Eurographics Assoc., 2007, pp. 35–44.
15. S. Tak and H.-S. Ko, "A Physically Based Motion Retargeting Filter," *ACM Trans. Graphics*, vol. 24, no. 1, 2005, pp. 98–117.
16. H.J. Shin, L. Kovar, and M. Gleicher, "Physical Touch-Up of Human Motions," *Proc. 11th Pacific Conf. Computer Graphics and Applications*, IEEE CS Press, 2003, pp. 194–203.
17. M.B. Popovic, A. Goswami, and H. Herr, "Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications," *Int'l J. Robotics Research*, vol. 24, no. 12, 2005, pp. 1013–1032.
18. A. Sulejmanpasić and J. Popović, "Adaptation of Performed Ballistic Motion," *ACM Trans. Graphics*, vol. 24, no. 1, 2005, pp. 165–179.
19. P.S.A. Reitsma and N.S. Pollard, "Perceptual Metrics for Character Animation: Sensitivity to Errors in Ballistic Motion," *ACM Trans. Graphics*, vol. 22, no. 3, 2003, pp. 537–542.
20. R. McDonnell, F. Newell, and C. O'Sullivan, "Smooth Movers: Perceptually Guided Human Motion Simulation," *Proc. 2007 ACM Siggraph/Eurographics Symp. Computer Animation* (SCA 07), Eurographics Assoc., 2007, pp. 259–269.

Ballistic Paths

Traditional kinematic animation systems feature manipulators that let animators easily create motion paths along the particular transformation's direction. For example, by specifying two keys along an x-translation, an animator can create a straight path in the x-direction. However, no straightforward way exists to create a ballistic path because creating it requires knowing

- an object's center of mass and
- a constraint, such as the starting velocity.

If we assume that no external forces affect the mass while in flight, we can describe the point mass's trajectory r with respect to time t as

$$\mathbf{r}(t) = \mathbf{a} + t\mathbf{b} + \frac{1}{2}Mt^2\mathbf{g}, \quad (1)$$

in which $\mathbf{a}$ and $\mathbf{b}$ are the two parameters determined from a ballistic motion's constraints, such as an origin and a destination, and traveling-time constraints. M is mass; $\mathbf{g}$ is the gravitational constant. The system creates ballistic paths in real time as animators move around the two endpoints and modify the duration time. Figure 1 shows screen captures of the ballistic paths being manipulated. To view multiple curves, users can vary the ballistic phase's duration.

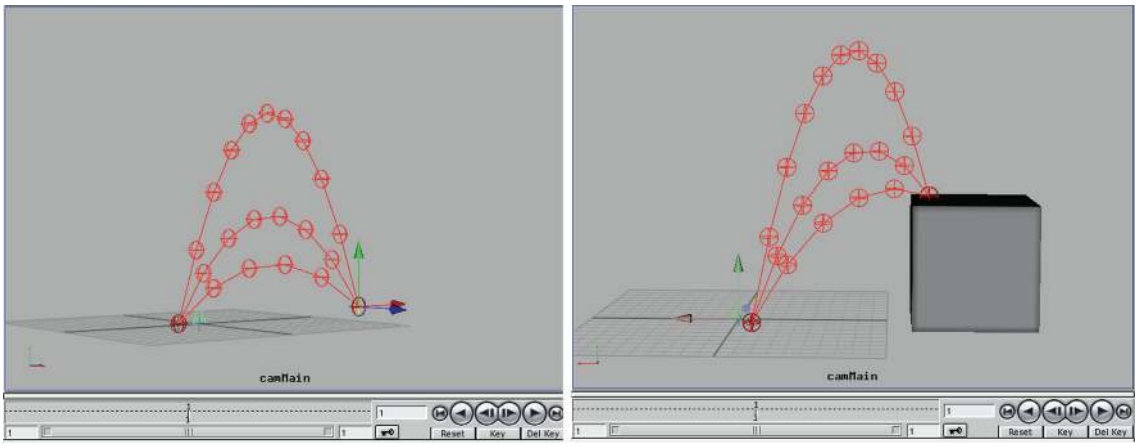
Animators can generate a physically plausible path by setting starting and ending points as constraints in 3D space. Because generating a ballistic path between two points is an underconstrained problem, animators can also generate multiple ballistic trajectories by using the minimum and maximum time span. Each curve presents a proper ballistic trajectory by indicating the path an object must follow to meet the location constraints at differing times. Animators can also create ballistic paths by specifying the first or last frame's position and velocity.

Using the ballistic paths, animators can easily correct linear momentum—and thus the center of momentum (COM) trajectory—of a character's ballistic motion. An animator chooses the desired ballistic path, then the system computes the character's COM at each frame and translates its root node so that the COM coincides with the ballistic path's current position. The resulting motion is physically correct in terms of linear momentum.

Animators can constrain a character's or an object's motion to a ballistic path by

- retaining the starting and ending locations and adjusting the timing to accommodate the ballistic path (see Figure 2), or
- manually adjusting the ballistic path until it mostly matches the original path and then altering the character's motion by retiming all the motion that occurs during the ballistic phase (see Figure 3).

When adapting a ballistic path to the original motion path, the animator visually modifies the



(a) (b)

Figure 1. Manipulating ballistic paths. (a) Our tool generates multiple ballistic paths between two locators. (b) Animators can use it to pregenerate paths between two points in a scene involving motion such as jumping or falling off a tall structure.

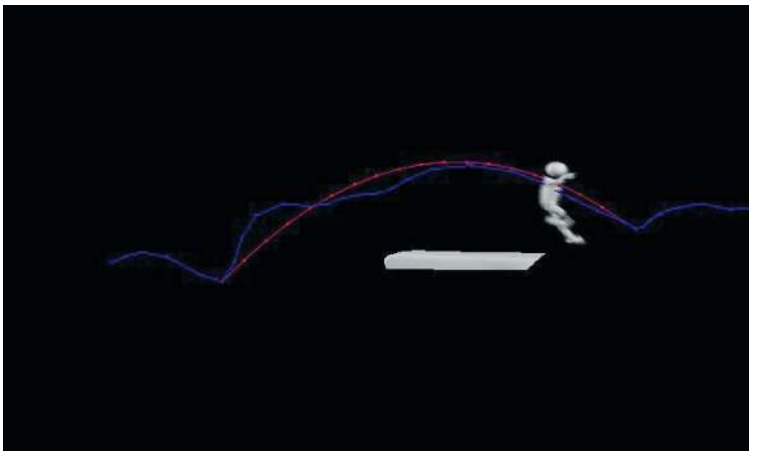


Figure 2. Matching a ballistic path. The blue curve is the trajectory of the center of mass of the character's (manually created) animation. Our system suggests the physically correct ballistic path (the red curve) that the character's center of mass should follow. The system lets animators automatically change the original animation to match physical laws.

ballistic path until he or she finds a good visual match.

We don't pay particular attention to the smoothness of the ballistic trajectory's start and end. Rather, we use the animator's original preparatory motion during the preballistic and postballistic phases. This doesn't introduce visual artifacts as long as the original motion doesn't differ much from the corrected motion, which is usually the case. When the artifacts are visible, animators can use conventional animation tools to modify the frames in the preballistic and postballistic phases during takeoff and landing.

The Angular-Momentum Tool

You can change a character's angular momentum in many ways. Our tool changes it by rotating the character's global orientation while keeping the original keyframe animation of each body part unchanged. By doing this, we can preserve the style of the animation that the animator carefully crafted while improving its physical realism. We provide an efficient algorithm to achieve this goal. With this algorithm, modification of angular momentum occurs online so that the animators get prompt visual feedback (see Figure 4).

For notational convenience, we use the generalized notations for the velocity, inertia, and momentum that provide the combined representation of the angular and linear properties. For the mathematical preliminaries of the generalized notations derived from Lie group theory, see the "Mathematical Preliminaries" sidebar.

Let $\mathbf{v}_i$, $\mathbf{J}_i$, and $\mathbf{h}_i$ denote the generalized velocity, inertia, and momentum of a body part i, with the body part 0 being the character's root. We can express a velocity $\mathbf{v}_i$ as the sum of the root's velocity and the relative velocity of i to the root:

$$\mathbf{v}_i = {}^{i}\mathbf{v}_0 + \mathbf{u}_i,$$

in which ${}^{i}\mathbf{v}_0$ is the velocity of the root expressed in i's body frame. (The left superscript indicates a symbol's reference frame. ${}^{0}\mathbf{v}_i$, ${}^{c}\mathbf{v}_i$, and ${}^{w}\mathbf{v}_i$ are $\mathbf{v}_i$ expressed in the root frame, the COM frame, and the world frame, respectively. We don't use the left superscript when the symbol is expressed in its own body frame. For example, $\mathbf{v}_i = {}^{i}\mathbf{v}_i$.)

Likewise, we can divide the generalized momentum into two parts:

$$\begin{aligned}\mathbf{h}_i &= \mathbf{J}_i\mathbf{v}_i \\ &= \mathbf{J}_i({}^{i}\mathbf{v}_0 + \mathbf{u}_i) \\ &= \mathbf{J}_i{}^{i}\mathbf{v}_0 + \mathbf{a}_i,\end{aligned}$$

in which $\mathbf{a}_i = \mathbf{J}_i\mathbf{u}_i$ represents the momentum induced by the velocity of i relative to the root. Then, we compute a character's generalized momentum in the root frame as

$$
\begin{aligned}
{}^0\mathbf{h} &= \sum_i {}^0\mathbf{h}_i \\
&= \sum_i {}^0\mathbf{J}_i\mathbf{v}_0 + {}^0\mathbf{a}_i \\
&= \left(\sum_i {}^0\mathbf{J}_i\right)\mathbf{v}_0 + \sum_i {}^0\mathbf{a}_i\,, \\
&:= {}^0\hat{\mathbf{J}}\mathbf{v}_0 + {}^0\mathbf{a}, \qquad (2)
\end{aligned}
$$

in which the composite rigid-body inertia $\hat{\mathbf{J}}$ denotes the aggregate inertia of the whole multibody system of a current configuration. ${}^0\hat{\mathbf{J}}$ and ${}^0\mathbf{a}$ are a function of joint angles only and are independent of the root's motion. So, they remain constant while we manipulate the root's translation and rotation. Using Equation 2, we can efficiently compute a character's momentum by calculating $\mathbf{v}_0$ instead of recalculating each body part's velocity as we manipulate the root.

We want to determine the configuration of the root $\mathbf{T}_0$ at a point in time at which the character has the user-specified momentum ${}^c\mathbf{h}^*$. We compute the character's velocity from the configurations of the current and previous time step. We modify only $\mathbf{T}_0$ at the current time step; we keep $\mathbf{T}_0$ at the previous time step fixed. Because we keep the linear momentum fixed, the COM frame (a coordinating frame that's parallel to the world frame with its origin coinciding with the COM) doesn't change while we rotate the character around the COM. So, given ${}^c\mathbf{h}^*$, its transformation with respect to the world frame ${}^w\mathbf{h}^* = \mathrm{Ad}^*_{\mathbf{C}^{-1}}\,{}^c\mathbf{h}^*$ is also constant while we manipulate the character. For convenience, we find $\mathbf{T}_0$ such that it creates ${}^w\mathbf{h}^*$.

The character's current momentum with respect to the world frame is

$$
{}^w\mathbf{h} = \mathrm{Ad}^*_{\mathbf{T}_0^{-1}}\,{}^0\mathbf{h}\,. \qquad (3)
$$

Both $\mathrm{Ad}^*_{\mathbf{T}_0^{-1}}$ and ${}^0\mathbf{h}$ are functions of $\mathbf{T}_0$, and a closed-form solution doesn't exist. So, we iteratively update $\mathbf{T}_0$ so that ${}^w\mathbf{h}$ approaches ${}^w\mathbf{h}^*$. Specifically, we update $\mathbf{T}_0$ by some $\mathbf{x}$ which is defined as

$$
\hat{\mathbf{x}} = \mathbf{T}_0^{-1}\delta\mathbf{T}_0\,,
$$

and find a suitable $\mathbf{x}$ that drives ${}^w\mathbf{h}$ to ${}^w\mathbf{h}^*$. To this end, we first relate the change of $\mathbf{v}_0$ with $\mathbf{x}$. From the definition $\hat{\mathbf{v}}_0 = \mathbf{T}_0^{-1}\dot{\mathbf{T}}_0$,

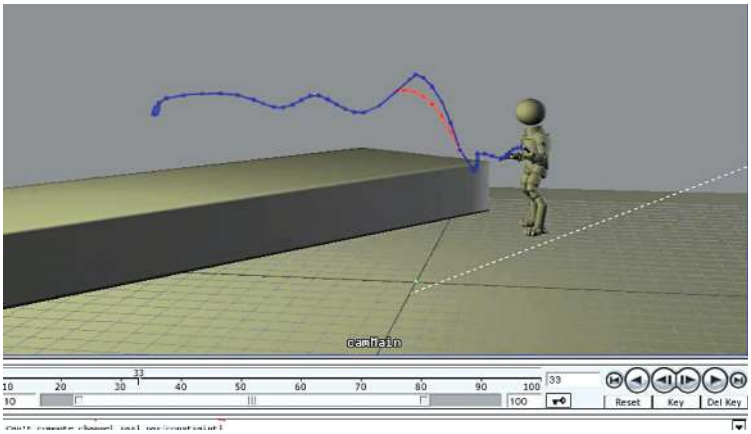


Figure 3. A character walking and jumping. The ballistic path (red) requires two more frames to complete the trajectory than does the animated path (blue). The keys for the ballistic path, which determine the timing, appear at slightly different locations than those in the animated path.

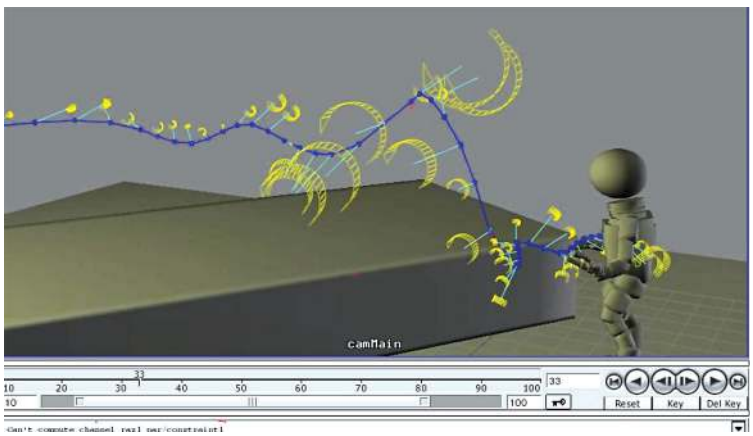


Figure 4. Visualization of angular momentum. The angular momentum appears as a vector protruding from the character's center of mass. The yellow arrows indicate the motion's direction (using the right-hand rule); the vectors' size indicates the relative amount of rotation about that axis.

$$
\begin{aligned}
\delta\hat{\mathbf{v}}_0 &= -\left(\mathbf{T}_0^{-1}\delta\mathbf{T}_0\mathbf{T}_0^{-1}\right)\dot{\mathbf{T}}_0 + \mathbf{T}_0^{-1}\delta\dot{\mathbf{T}}_0 \\
&= -\hat{\mathbf{x}}\hat{\mathbf{v}}_0 + \mathbf{T}_0^{-1}\frac{d}{dt}\left(\mathbf{T}_0\hat{\mathbf{x}}\right) \\
&= -\hat{\mathbf{x}}\hat{\mathbf{v}}_0 + \mathbf{T}_0^{-1}\left(\dot{\mathbf{T}}_0\hat{\mathbf{x}} + \mathbf{T}_0\dot{\hat{\mathbf{x}}}\right) \\
&= -\hat{\mathbf{x}}\hat{\mathbf{v}}_0 + \hat{\mathbf{v}}_0\hat{\mathbf{x}} + \dot{\hat{\mathbf{x}}} \\
&\approx \mathrm{ad}_{\hat{\mathbf{v}}_0}\hat{\mathbf{x}} + \frac{\hat{\mathbf{x}}}{h}\,, \qquad (4)
\end{aligned}
$$

in which h is the time step. Using Equation 4 and recalling that ${}^0\hat{\mathbf{J}}$ and ${}^0\mathbf{a}$ are constant, we express $\delta{}^0\mathbf{h}$ in terms of $\mathbf{x}$:

$$
\begin{aligned}
\delta{}^0\mathbf{h} &= {}^0\hat{\mathbf{J}}\delta\mathbf{v} \\
&\approx {}^0\hat{\mathbf{J}}\left(\mathrm{ad}_{\mathbf{v}_0}\mathbf{x} + \frac{\mathbf{x}}{h}\right).
\end{aligned}
$$

Mathematical Preliminaries

Given a homogeneous representation of a moving body frame $\mathbf{T} = (\mathbf{R}, \mathbf{p})$ in which $\mathbf{R} \in \mathbb{R}^{3\times3}$ denotes rotation and $\mathbf{p} \in \mathbb{R}^3$ translation, its *generalized velocity* $\mathbf{v} = [\omega^T, v^T]^T$ expressed in the instantaneous body frame (hence dubbed *body velocity*) is

$$\hat{\mathbf{v}} = \mathbf{T}^{-1}\dot{\mathbf{T}} = \begin{bmatrix} [\omega] & v \\ \mathbf{0} & 0 \end{bmatrix}, \quad \text{(A)}$$

in which ω and v are, respectively, the angular and linear velocities of $\mathbf{T}$ expressed in the body frame. $[\omega]$ is the skew-symmetric matrix of ω; that is, $[\omega]\eta = \omega \times \eta$ for any vector $\eta \in \mathbb{R}^3$. We use $\hat{\mathbf{v}}$ for the 4 × 4 matrix representation of $\mathbf{v}$. The generalized momentum $\mathbf{h} = [\mathbf{k}^T, \mathbf{l}^T]^T$ is expressed as

$$\mathbf{h} = \mathbf{J}\mathbf{v}, \quad \text{(B)}$$

in which $\mathbf{k} \in \mathbb{R}^3$ and $\mathbf{l} \in \mathbb{R}^3$ represent the angular and linear momentum (with respect to the body frame), respectively. The rigid body's *generalized inertia* $\mathbf{J} \in \mathbb{R}^{6\times6}$ has this structure:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & m[\mathbf{r}] \\ m[\mathbf{r}]^T & m\mathbf{1} \end{bmatrix},$$

in which m is the mass, $\mathbf{I} \in \mathbb{R}^{3\times3}$ is the rotational inertia matrix, $\mathbf{r} \in \mathbb{R}^3$ is the position of the center of mass, and $\mathbf{1} \in \mathbb{R}^{3\times3}$ is the identity matrix. Equation B is coordinate-invariant (it holds with respect to any coordinate frame).

Given a coordinate frame $\mathbf{T}$ and a generalized velocity $\mathbf{g} = [\omega^T, v^T]^T$, the adjoint mapping $\mathrm{Ad}_\mathbf{T}$ is represented as $\mathrm{Ad}_\mathbf{T}\hat{\mathbf{g}} = \mathbf{T}\hat{\mathbf{g}}\mathbf{T}^{-1}$, or in matrix form as

$$\mathrm{Ad}_\mathbf{T}\mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ [\mathbf{p}]\mathbf{R} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix}.$$

We use the adjoint mapping in the coordinate transformation of the generalized velocity. The corresponding dual adjoint mapping that performs the coordinate transformation of the generalized momentum is $\mathrm{Ad}^*_\mathbf{T}$; it has the form of the transpose of $\mathrm{Ad}_\mathbf{T}$; that is, $\mathrm{Ad}^*_\mathbf{T} = \mathrm{Ad}^T_\mathbf{T}$. For example, the generalized velocity, momentum, and inertia with respect to the world frame (${}^w\mathbf{v}$, ${}^w\mathbf{h}$, ${}^w\mathbf{J}$, respectively, with the left superscript w implying the "world" reference frame) are

$${}^w\mathbf{v} = \mathrm{Ad}_\mathbf{T}\mathbf{v}$$

$${}^w\mathbf{h} = \mathrm{Ad}^*_{\mathbf{T}^{-1}}\mathbf{h}$$

$${}^w\mathbf{J} = \mathrm{Ad}^*_{\mathbf{T}^{-1}}\mathbf{J}\mathrm{Ad}_{\mathbf{T}^{-1}}.$$

We can easily verify that $\mathrm{Ad}^{-1}_\mathbf{T} = \mathrm{Ad}_{\mathbf{T}^{-1}}$ and $\mathrm{Ad}_{\mathbf{T}_1}\mathrm{Ad}_{\mathbf{T}_2} = \mathrm{Ad}_{\mathbf{T}_1\mathbf{T}_2}$.

Assuming that link 0 of a multibody system is the root link, the configuration $\mathbf{T}_i$ of the body frame $\{i\}$ of i with respect to the world frame is

$$\mathbf{T}_i = \mathbf{T}_0\mathbf{G}_i, \quad \text{(C)}$$

in which $\mathbf{T}_0$ is the configuration of the root and $\mathbf{G}_i$ denotes the relative configuration of $\{i\}$ with respect to the root. Substituting Equation C into Equation A, we can decompose the body velocity $\mathbf{v}_i$:

$$\mathbf{v}_i = {}^i\mathbf{v}_0 + \mathbf{u}_i,$$

in which ${}^i\mathbf{v}_0 = \mathrm{Ad}_{\mathbf{G}_i^{-1}}\mathbf{v}_0$ is the velocity of the root expressed in the body frame $\{i\}$ and $\mathbf{u}_i = \left[\omega_u^T, v_u^T\right]$ is the relative velocity of i to the root:

$$\begin{bmatrix} [\omega_u] & v_u \\ \mathbf{0} & 0 \end{bmatrix} = \mathbf{G}_i^{-1}\dot{\mathbf{G}}_i.$$

The left superscript denotes the reference frame. In the main article, we use the transformations of the generalized velocity, momentum, and inertia of a link i to the root: ${}^0\mathbf{v}_i = \mathrm{Ad}_{\mathbf{G}_i}\mathbf{v}_i$, ${}^0\mathbf{h}_i = \mathrm{Ad}^*_{\mathbf{G}_i^{-1}}\mathbf{h}_i$, and ${}^0\mathbf{J}_i = \mathrm{Ad}^*_{\mathbf{G}_i^{-1}}\mathbf{J}_i\mathrm{Ad}_{\mathbf{G}_i^{-1}}$.

The Lie bracket $\mathrm{ad}_\mathbf{g}$ is another mapping for the generalized velocity, defined as $\mathrm{ad}_{\hat{\mathbf{g}}_1}\hat{\mathbf{g}}_2 = \hat{\mathbf{g}}_1\hat{\mathbf{g}}_2 - \hat{\mathbf{g}}_2\hat{\mathbf{g}}_1$ or, in matrix form,

$$\mathrm{ad}_{\mathbf{g}_1}\mathbf{g}_2 = \begin{bmatrix} [\omega_1] & \mathbf{0} \\ [v_1] & [\omega_1] \end{bmatrix} \begin{bmatrix} \omega_2 \\ v_2 \end{bmatrix}.$$

The dual $\mathrm{ad}^*_\mathbf{g}$ for the generalized momentum is its transpose $\mathrm{ad}^*_\mathbf{g} = \mathrm{ad}^T_\mathbf{g}$. Note that $\mathrm{ad}_\mathbf{g}\mathbf{v} = -\mathrm{ad}_\mathbf{v}\mathbf{g}$ and $\mathrm{ad}_\mathbf{g} + \mathrm{ad}_\mathbf{v} = \mathrm{ad}_{\mathbf{g}+\mathbf{v}}$. The Lie bracket occurs when Ad is differentiated. For example, if $\mathrm{Ad}_\mathbf{T}$ is differentiated with respect to time t,

$$\frac{d}{dt}\mathrm{Ad}_\mathbf{T} = \mathrm{Ad}_\mathbf{T}\mathrm{ad}_\mathbf{v},$$

in which $\mathbf{v}$ is the body velocity of $\mathbf{T}$. For the proof, see "Newton-Type Algorithms for Dynamics-Based Robot Movement Optimization."[1]

Reference

1. S.-H. Lee et al., "Newton-Type Algorithms for Dynamics-Based Robot Movement Optimization," *IEEE Trans. Robotics*, vol. 21, no. 4, 2005, pp. 657–667.

Finally, we compute the change of ${}^{w}\mathbf{h}$ due to $\mathbf{x}$:

$$\begin{aligned}\delta\,{}^{w}\mathbf{h} &= \delta\left(\mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\right){}^{0}\mathbf{h} + \mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\delta\,{}^{0}\mathbf{h}\\ &= -\mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\mathrm{ad}^{*}_{\mathbf{x}}\,{}^{0}\mathbf{h} + \mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\delta\,{}^{0}\mathbf{h}\\ &= \mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\mathrm{ad}^{*}_{{}^{0}\mathbf{h}}\mathbf{x} + \mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\delta\,{}^{0}\mathbf{h}\\ &\approx \mathrm{Ad}^{*}_{\mathbf{T}_0^{-1}}\left(\mathrm{ad}^{*}_{{}^{0}\mathbf{h}} + {}^{0}\hat{\mathbf{J}}\mathrm{ad}_{\mathbf{v}_0} + \frac{1}{h}\,{}^{0}\hat{\mathbf{J}}\right)\mathbf{x}, \end{aligned} \quad (5)$$

in which $\mathrm{ad}^{*}_{{}^{0}\mathbf{h}}\mathbf{x}$ and ${}^{0}\hat{\mathbf{J}}\mathrm{ad}_{\mathbf{v}_0}\mathbf{x}$ account for the effect of the coordinate change of $\mathbf{T}_0$ due to $\mathbf{x}$ on ${}^{0}\mathbf{h}$ and $\mathbf{v}_0$, respectively; $1/h\,{}^{0}\hat{\mathbf{J}}\mathbf{x}$ is the added momentum due to $\mathbf{x}$. By solving Equation 5 for $\mathbf{x}$, we can compute the $\mathbf{x}$ that will create the desired $\delta{}^{w}\mathbf{h}$.

Based on the relations we just derived, here's the algorithm to compute $\mathbf{T}_0$ given ${}^{c}\mathbf{h}^{*}$:

1. ${}^{w}\mathbf{h}^{*} = \mathrm{Ad}^{*}_{\mathbf{C}^{-1}}\,{}^{c}\mathbf{h}^{*}$, in which $\mathbf{C}$ is the COM frame with respect to the world frame.
2. Compute ${}^{0}\hat{\mathbf{J}}$ and ${}^{0}\mathbf{a}$ (see Equation 2).
3. Compute ${}^{w}\mathbf{h}$ (see Equation 3).
4. **while** $\delta{}^{w}\mathbf{h} = {}^{w}\mathbf{h}^{*} - {}^{w}\mathbf{h}$ is above a threshold, **do**
5. Compute $\mathbf{x}$ by solving Equation 5.
6. $\mathbf{T}_0 \leftarrow \mathbf{T}_0 e^{\gamma\hat{\mathbf{x}}}$, in which $0 < \gamma \leq 1$.
7. Update $\mathbf{v}_0$, ${}^{0}\mathbf{h}$ (see Equation 2), and ${}^{w}\mathbf{h}$.

γ controls the distance for the next iteration. We change only the angular part of ${}^{c}\mathbf{h}^{*}$ and keep the linear momentum fixed. However, both the angular and linear parts of ${}^{w}\mathbf{h}^{*}$ change, owing to the coordinate transformation.

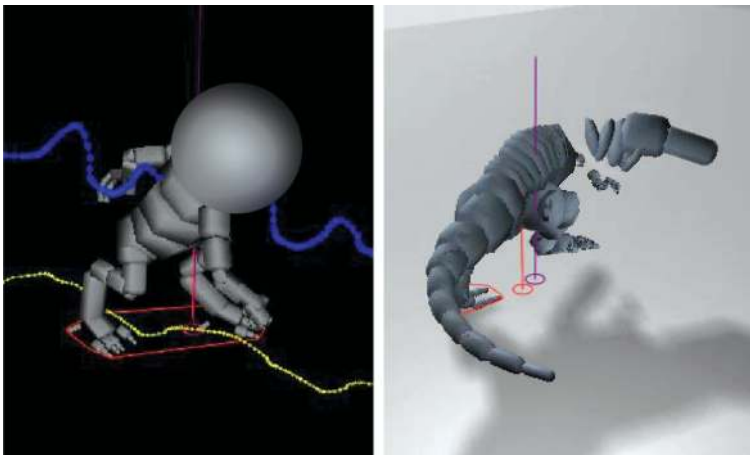
Keyframe Animations' Physical Accuracy

Here, we investigate the physical accuracy of professional animators' keyframe animations. Specifically, we compute the ballistic motions' center of mass, linear momentum, and angular momentum and investigate how accurately these properties follow Newton's laws. For walking and running animations, we compute the center of pressure (COP) and verify whether it is actually in the support polygon. Figure 5 shows snapshots in which COM and COP are visualized.

Ballistic Motions

We collected 13 ballistic motions of human-like characters, such as jumping and falling motions. Each motion lasted from 4 to 27 frames. We examined 150 frames total.

In the ballistic phase, the path of the COM that's projected to the horizontal plane should form a straight line. Figure 6a is the histogram of errors in the ballistic motions' horizontal plane.



(a) **(b)**

Figure 5. The center of mass projected on the ground (red circle), the support polygon (red polygon), and the center of pressure (COP; the purple circle) for keyframe animation of (a) a humanoid character and (b) a nonhumanoid character. The nonhumanoid character's horizontal shape causes greater instability in the COP calculation than the humanoid character's shape does.

Here, the error is the distance from the COM's horizontal projection to the line segment connecting the horizontal projection of the COMs of the first and last frames. The errors are normalized by the line segment's length. The figure shows that the error is less than 0.1 for more than 90 percent of the frames. This relatively low error seems due to how animators create keyframes for ballistic motions. Because animators usually create a straight line from the root node's starting and ending position and adjust the heights of the in-between keyframes, the COM trajectory's horizontal projection shows a mostly straight line unless a character often changes its pose.

Figures 6b and 6c show the histogram of normalized errors of the linear momentum in the horizontal (x-z) plane. Here, the error is the difference between the current momentum and the average momentum, normalized by the average momentum's magnitude. When comparing the error of the linear momentum with that of the COM, we can see that the keyframe's location in the line is less correct from the perspective of the linear momentum, even though the in-between keyframes make a line that's mostly straight.

Figures 6d, 6e, and 6f show the histogram of normalized errors of the angular momentum in the x, y, and z directions. Here, we define the error in the same way as in the linear momentum. The error in the angular momentum is much greater than that in the linear momentum. One reason for this might be that angular momentum is less perceptible to human eyes than linear momentum is. Except for flipping jumps, during which

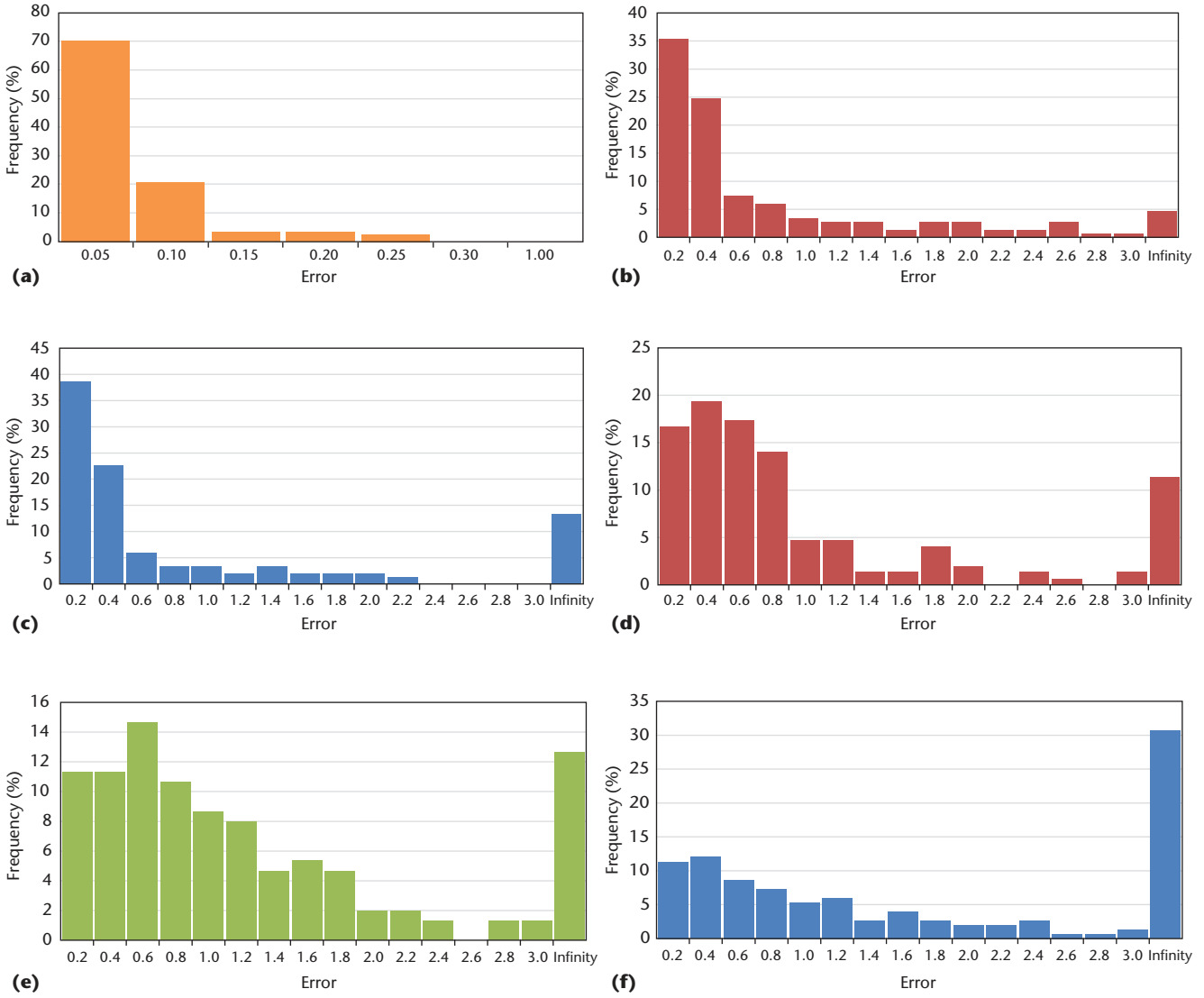


Figure 6. Measuring the physical correctness of ballistic motions created by professional animators using conventional keyframing. The normalized error of the (a) center of momentum (COM) in the horizontal plane, (b) linear momentum in the *x* direction, (c) linear momentum in the *z* direction, (d) angular momentum in the *x* direction, (e) angular momentum in the *y* direction, and (f) angular momentum in the *z* direction.

the whole body's rotation is explicit, the ballistic motion's angular momentum is hard to perceive without a visualization tool such as ours.

By fitting Equation 1 to the COM trajectory, we estimate a scene's gravity. Table I shows this gravity relative to earth gravity (9.8 m/s^2) for each scene. The gravity deviates considerably among scenes ranging from 0.5 g to 3.3 g. The median of the gravities is 0.95 g, which is close to real gravity. This shows that the keyframe animations create ballistic motions under true gravity on average, but that large deviations exist among scenes.

Research has shown that observers can detect deviations in horizontal or vertical accelerations.[5] Our data demonstrates that hand-animated motion often exceeds such thresholds. Another interpretation of the variation in estimated gravity is that big differences exist in animators' notions of a character's size. If that's true, lower gravity in the data indicates the animator consid-

Table 1. Gravity in the test scenes.

	Scene												
	1	2	3	4	5	6	7	8	9	10	11	12	13
g*	1.02	0.89	2.20	0.47	0.50	1.52	1.41	0.77	3.35	1.81	0.43	0.56	2.51

**g: measured gravity/9.81.*

ers the character bigger than it actually is; higher gravity means the animator thinks the character is smaller than it is. Other research has suggested that preparatory motion might also be an important consideration in the perceived visual quality of ballistic motion.[6] Our method doesn't address the preparatory or recovery phases; it leaves such tasks to animators.

Ambulatory Motions

We collected five scenes of walking motions (165 frames) and five scenes of running motions (113 frames) of biped characters and examined whether the COP is in the support polygon. Figure 7 shows the histogram of the COP's normalized distance from the support polygon. The distance is positive when the COP is outside the support polygon; it's normalized by the support polygon's mean radius (computed as the perimeter divided by 2π).

For walking, the COP is in the support polygon in more than 60 percent of the frames and is within the normalized distance of 1 in 85 percent of frames. The experiment shows that the keyframe animation has relatively high accuracy in terms of the COP. This suggests that the COP is an important indicator of ambulatory motions' realism. Somewhat naturally, the error grows as a character's speed increases. For running, the COP is in the support polygon in only 33 percent of frames and is within the normalized distance of 1 in 70 percent of frames.

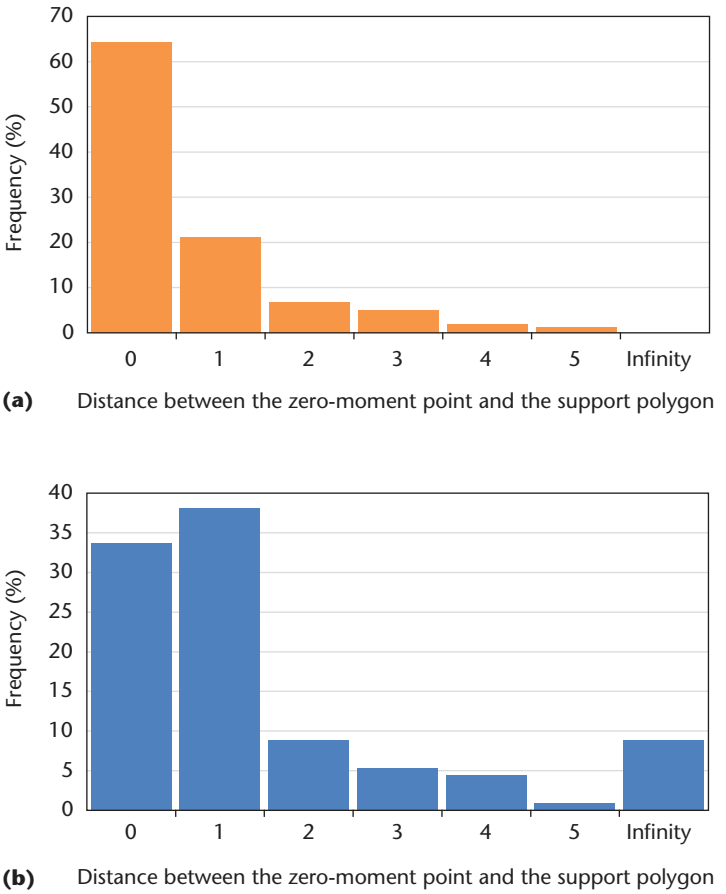


Figure 7. Histograms of the normalized distance between the COP and the support polygon for animations of (a) walking and (b) running created by professional animators using conventional keyframing. The COPs of keyframe animation turned out to be fairly accurate, being mostly within the normalized distance of 1 from the support polygon.

Discussion

Here, we discuss the advantages and limitations of our system and observations made when the system was used by animators.

Effectiveness and Impact

Many animators have found our system useful for improving keyframe animation's realism. The manually created animations in Figure 2 and the accompanying video (see http://doi.ieeecomputersociety.org/10.1109/MCG.2010.22) are the results of substantial time and effort devoted by professional animators. So, some of these animations are already quite realistic and leave less room for improvement using our tool. In some sense, this shows that physical realism is a key factor for high-quality animation. Other examples show that our system improves the physical realism significantly. Also, when our system is used from the beginning of the animation process, it might increase that process's efficiency.

Our visualization tools can also serve as a "gavel"—that is, confirmation that animation is indeed realistic. Individual animators might have their own sense of physical correctness. This can cause debate over how a character should move when collaborating animators have a different sense of physical intuition. We've observed that our tool helps animators agree on physically correct animation by quantifying the motion's discrepancy. For example, the ballistic path can indicate the exact number of frames in which an animation should be slowed down or sped up.

We determined that in a character-heavy live-action film, between 10 and 16 percent of the shots using animated characters have ballistic motion such as falling or jumping. Our system could improve many of these shots. Of course, many more shots benefit from COM and COP visualization.

Our system is most effective when the animator understands how to change the physical curves to create better animations. So, it was useful to create videos that showed, for example, the proper location of the COM and COP (or zero-moment point) during walking, running, or other motions.

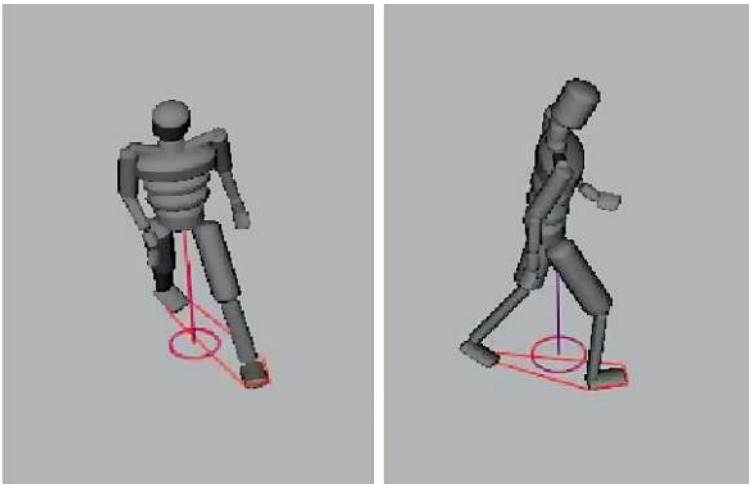


Figure 8. Using motion capture data as a training tool for animators. The COM is outside the support polygon while the character turns.

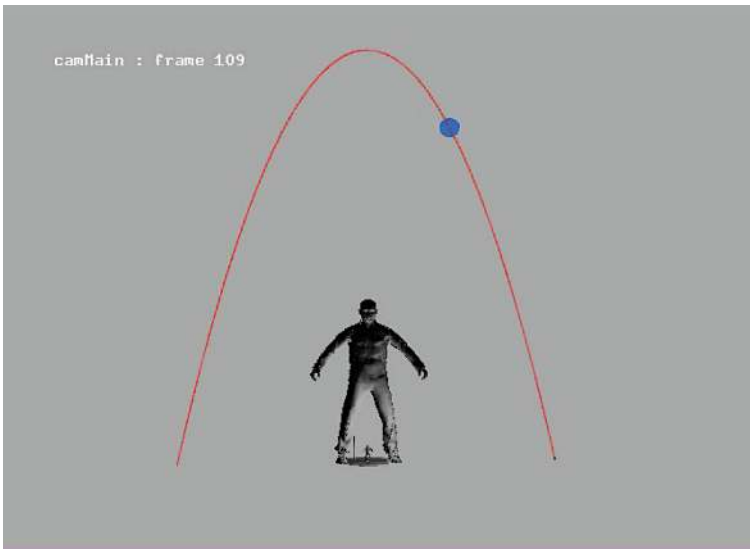


Figure 9. How scale affects the appearance of motion. The figure shows a normal-sized man and a man 10 times larger. A ball is placed in motion around each of them. The ball moving around the larger man will take 3.19 times as long as the one moving around the normal-sized man.

Such training videos can effectively teach animators how to interpret our system, as in Figure 8. Of course, the animator chooses whether to make the character act human-like.

Use of COM and COP

The COM and COP guidelines explicitly expressed the notions of physical correctness in animations. For example, a humanoid's COM tends to follow a path from about the contact foot to the other contact foot at a later time. Our system was able to serve as an instructional aid for creating smoother motion because, for example, users could directly see the idea of "carrying the weight from one foot to another" in the COM's projection.

Scaling Large and Small Characters

The characters' scale can dramatically impact the resulting animation. Other research has provided detailed explanations for scaling dynamic control systems on the basis of time, positions, and velocities.[7] We relate such scaling strategies to the application of kinematically based animation of large and small characters. For example, large creatures appear to move more slowly under proper physical conditions, whereas small creatures appear to move more quickly. The time t required for an object to travel to a maximum height h is

$$t = \sqrt{2h/g}\,, \tag{22}$$

in which g is the force of gravity. So, gravity-related movements such as walking (in which gravity pulls the character's swinging leg to the ground) or throwing objects will appear slower or faster if we consider two characters of differing sizes, h_1 and h_2, and their respective times, t_1 and t_2, to complete such actions. The ratio of time required for that action to occur would be $t_1/t_2 = \sqrt{h_1/h_2}$.

Our observations have shown that animators often make large errors when the characters they're animating are either much larger or much smaller than normal-sized objects. For example, giant men will move too quickly for their relative size, and small men will move much too slowly for their real size. We hypothesize that this is due to the familiarity of seeing characters of normal size move and the unfamiliarity of seeing giants or miniature people move (see Figure 9).

Limitations and Future Work

Our method for modifying angular momentum changes only the character's global orientation. It doesn't change the character's pose. There are many other ways to correct the angular momentum that involve changing parts of the character's body while leaving other parts the same. For example, a person can change his or her angular momentum by windmilling his or her arms in a circular manner. We don't provide a tool to let animators explore all these possibilities. Instead, we focus on ease of use and automation. It isn't clear that a single useful correction method exists that will yield better animation due to animation constraints, such as requiring a character's feet to be in contact with the ground during landing. So, it's the animator's role to manually correct the character's posture to resolve this discrepancy. In the previous example, the angular momentum during flight can be automatically adjusted but would

need to be manually smoothed with the landing posture to obtain correctness.

Also, here we computed the COP under the assumption that the ground is flat. However, characters often walk or run on uneven ground. When the contact points between the character and the ground aren't coplanar, our system can't define the support polygon. For such cases, we'll need to extend the support polygon to 3D space.

The algorithms we use to generate physical visualization are straightforward to implement and compatible with most kinematic animation systems. In addition, the system doesn't require animators to change the methods by which they generate animation, thus leveraging their existing skills. Also, the generated motion's quality is additive—it doesn't replace the underlying animation. The animator may choose to use the system only if it enhances the motion's realism.

Although we designed our system for live-action visual effects, the techniques work for almost any animation purpose that requires or desires better physical realism for characters, such as systems for fully 3D environments or prebaked animations for video games.

Our system can also help enforce consistency of character motion across an animation studio. Typically, several animators will create animations of a particular character for different scenes. A movie requiring heavy visual effects might require coordinating dozens of animators to produce hundreds of animations for a small number of characters. With our system, different animators' animations of the same character tend to be more consistent with each other. This is because the animators don't have to rely solely on their individual senses of timing and space; they can use our system's interactive visual feedback.

Although we designed our system for live-action visual effects, the techniques work for almost any animation purpose that requires or desires better physical realism.

Acknowledgments

This material is based mostly on research we completed while at Rhythm & Hues Studios. Both of us are corresponding authors. We thank the anonymous reviewers for their helpful comments, which improved the article. Sung-Hee Lee was supported partly by the Global Frontier R&D Program of the National Research Foundation, Korea (NRF-M1AXA003-20100029751).

References

1. Z. Kačić-Alesić, M. Nordenstam, and D. Bullock, "A Practical Dynamics System," Proc. 2003 *ACM Siggraph/Eurographics Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 7–16.
2. "Massive," Massive Software, 2011; www.massivesoftware.com.
3. "Havok Behavior," Havok Inc., 2011; www.havok.com/index.php?page=havok-behavior.
4. "Endorphin 2.7," NaturalMotion Ltd., 2011; www.naturalmotion.com/endorphin.
5. P.S.A. Reitsma and N.S. Pollard, "Perceptual Metrics for Character Animation: Sensitivity to Errors in Ballistic Motion," *ACM Trans. Graphics*, vol. 22, no. 3, 2003, pp. 537–542.
6. P.S.A. Reitsma, J. Andrews, and N.S. Pollard, "Effect of Character Animacy and Preparatory Motion on Perceptual Magnitude of Errors in Ballistic Motion," *Computer Graphics Forum*, vol. 27, no. 2, 2008, pp. 201–210.
7. N. Pollard, "Simple Machines for Scaling Human Motion," *Proc. Computer Animation and Simulation '99*, Springer, 1999, pp. 3–11.

Ari Shapiro *is a research scientist at the Institute for Creative Technologies. He previously worked as a graphics scientist at Rhythm & Hues Studios, and an R&D engineer at Industrial Light and Magic. His research interests include realistic character motion, physics-based animation, and animation tools. Shapiro has a PhD in computer science from the University of California, Los Angeles. Contact him at shapiro@ict.usc.edu.*

Sung-Hee Lee *is a lecturer at the Gwangju Institute of Science and Technology's School of Information and Communications. His research interests include human modeling and animation, physics-based animation, multibody dynamics, and humanoid robotics. He previously was a postdoctoral researcher at Honda Research Institute USA. Lee has a PhD in computer science from the University of California, Los Angeles. He's a member of IEEE and ACM Siggraph. Contact him at shl@gist.ac.kr.*