Practical Character Physics For Animators

Ari Shapiro and Sung Hee Lee

Abstract—We describe a graphics system that significantly improves the visual quality of certain types of 3D character motion animated through traditional means by inferring physical properties and correcting the results through the use of dynamics. These physical characteristics are visualized and provide information not normally available to traditional 3D animators, such as displaying the center of mass, angular momentum and zero moment point. By comparing the original path as generated by an animator, against a proper physically-based path generated by our tool, the animator is able to interactively modify the original motion path to more closely match the generated physics-based path. This often results in better quality character motion. Two different types of motion can be adjusted: animations which involve ballistic paths, such as falling and jumping, as well as animations involving character movement which require balance and posture, such as walking or running. This dynamics visualization method is integrated into a professional software system for use in a visual effects studio that incorporates live-action with 3D animated characters in feature film production. Our research shows that between 10% and 16% of the shots of a character-heavy feature film will incorporate ballistic motions that may be improved by our system.

Index Terms—physical simulation, character animation

INTRODUCTION

Given that one of the major advantages of 3D animation is its ability to make physically unrealistic scenarios convincing, physical realism is not an absolute principle that must be kept in all the 3D animation works. For example, an audience pays less attention to the physical realism of cartoon-like 3D characters than to photorealistic characters. However, realism is very important to the production of live-action visual effects when animated characters occupy the same scene as the live actors and the live environment. In such a scenario, a virtual character's movements must visually match the behavior and movements of the live environment, else the discrepancy will be obvious to the viewer. For example, a character jumping in a scene and thus being brought to the ground by gravity, must visually match an object being thrown in the same scene under the same gravitational force.

However, the traditional tools used to create the 3D character animation do not include dynamical information, leaving the dynamics of character motion to exist only implicitly within the animation framework. Some animation systems incorporate physical simulation of rigid and non-rigid solids, fluids, gases, and characters [1], [2], [3], [4]. In addition, technical animators will often apply dynamic effects in a post-processing step, such as creating the secondary-motion effects of muscular bulging and hair bouncing. However, the vast majority of character animators will create most of the character movement through kinematic means. The requirements for character animation in feature films often requires a fine-grained level of control over all parts of the character's movement that currently cannot be achieved by the state of art character dynamics simulation methods. In addition, the constraints of a particular shot might require that an animated character's motion violates the laws of physics. For example, a character that needs to move unnaturally in order to stay within a camera's view. Also, animators are typically trained using kinematic tools and thus develop a high level of proficiency using such tools.

Therefore, the conventional keyframing animation process has remained the method of choice for animation studios. In order to generate realistic-looking motion, a professional animator will typically use a combination of methods such as keyframing, inverse kinematics and other traditional tools such as curve editors. Animators also frequently employ the use of reference motion, such as videos of people and creatures performing a motion that they need to replicate. These references assist the animator in approximating the dynamics of motion, since traditional rigs do not include dynamical aspects such as masses and moments of inertia. Thus, physical plausibility must be generated without the direct input of these physical aspects.

In this paper, we describe an implementation of an interactive system that assists an animator in creating more physically plausible character motions as part of the animation process. To this end, our system allows an animator to view the motion of a character or object as if it were to obey the laws of physics. Specifically, the system produces visualizations of dynamical properties, such as the center of mass, momentum and balance. For example, our system creates a physically-

Ari Shapiro is with Rhythm & Hues Studios, Los Angeles, CA USA email: (ashapiro@rhythm.com).

Sung-Hee Lee is with Honda Research Labs, Mountain View, CA USA email: (sungheelee@gmail.com).

accurate ballistic motion path alongside the original kinematic path. By comparing the two paths, as well as viewing the additional dynamical information, the animator is then able to adjust the original animation towards 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, enabling non-animators to quickly correct existing animation without an animator's input.

The main purpose of system is not to automatically generate physically correct motion, but rather to inform the animator of the changes that would need to be made in order to make the motion physically correct. Since the animator ultimately has complete control over the extent to which their original animation is changed, the system is easily adoptable to a professional animator's toolset. We describe examples of using such a system for both ballistic motion as well as ambulatory (walking, running) motion.

Our research shows that professional animators often create motion that violates physical laws and such motions can be significantly improved by applying physical correctness.

The contributions of this paper are as follows. First, the incorporation of physical properties into an existing key frame-based system is a new contribution and does not currently exist in any commercial software package or studio animation system. To our knowledge, no such description has been published. We believe that this system demonstrates the capability to create an industry-wide improvement to existing animation systems and animator techniques due to the ease of implementation and the effective increase in animation realism for the shots that can be enhanced by this process.

Second, we develop tools that help animators modify ballistic motions of characters to enhance the physical realism, which is difficult to achieve using traditional key frame-based means. To this end, we create a ballistic path tool which corrects the linear momentum of an existing motion as well as creates new ballistic motions. We also develop an algorithm that let animator easily modify the angular momentum of a character.

Lastly, we investigate the physical accuracy of the high quality animations that are manually created by professional animators by measuring some important physical properties. Specifically, we compute the center of mass and momentum of the ballistic motions and investigate how accurately these properties follow Newton's laws. For walking and running animations, we compute the center of pressure and measure how far it falls from the support polygon. Our study shows the capabilities and limitations of the conventional kinematic

animation process in terms of physical realism and the result reported in this paper may provide useful data for studying human perception of the physics in character animation. To our knowledge, this paper is the first to report the physical accuracy of character animation manually created by skilled animators using key frame-based authoring tools.

RELATED WORK

The focus of this paper is assisting animators in creating physically realistic character animation by visualizing dynamical properties of keyframed animation. In contrast, most of physics-based animation techniques deal with automatic generation of animation with minimal user inputs. The space-time optimization approaches [5], [6] automatically create physically plausible animation by solving optimization problem subject to physical constraints as well as other constraints. Another approach has been to develop algorithms that control a character under physical simulation. Such dynamic controllers have been constructed to allow characters to perform simple athletic maneuvers [7], swimming [8], stable walking cycles [9], reactive motions such as falling [10] as well as other motions such as breathing and grasping [11], [12]. Such approaches can also be used in concert with kinematic animation [13] and motion capture [14]. Physics has also been used to create realistic secondary motions via physical simulation [15], [16].

These physics-based techniques pursue a very promising way of animation production, but they are yet to be widely employed by industry due to the aforementioned reasons. Also, physics-based animation is often computationally heavy, which prevents it from being used in an interactive authoring environment. Instead, our approach is developed to improve the physical realism of the conventional key framing-based techniques by visualizing dynamic properties of kinematic animation. Hence our system can be readily employed by industry.

In this paper, we visualize physical properties such as center of mass, momentum, and the center of pressure (COP). These properties has been explicitly used to increase physical realism of existing animation. [17] created flip-jumping motions using motion capture data in such a way that resulting motion obeys momentum preservation laws. [18] and [19] modified existing animation to increase physical realism by enforcing zero moment point (ZMP) constraint for locomotion or linear/angular momentum constraint for ballistic motions.¹ [21] proposed a new method for creating a physically valid ballistic motion by adapting an existing motion to new constraints.

¹COP and ZMP coincide for locomotion on the flat ground [20].

Despite the usefulness of these techniques, many animators still prefer manual editing of the motions because they need to control various aspects of the motion, ranging kinematic constraints to overall style of the animation. Our approach is different from their papers in that, again, we are primarily interested in enabling animators to create physically plausible animation along with various other requirements that they need to satisfy, rather than actually creating such motions.

Other research has shown that viewers can be sensitive to certain types of errors in ballistic motion [22], [23]. Our user studies show that animators often generate animations that exceed these thresholds, thus creating perceptibly implausible motion.

IMPROVING PHYSICAL REALISM

We found that simply visualizing the physical properties helps animators create more realistic animation. In addition, we develop a ballistic path tool that let an animator easily create or modify ballistic animation, and an angular momentum tool that rotates the global orientation of a character to achieve desired angular momentum. These features are integrated into keyframing-based animation authoring software.

Ballistic Paths

Traditional kinematic animation systems feature manipulators that allow easy creation of motion paths along the direction of the particular transformation. For example, by specifying two keys along an x-translation, an animator can create a straight path along in the x-direction. However, there is no straightforward way to create a ballistic path, since creation of such a path requires knowing: 1) the center of mass of an object, and 2) a constraint, such as the starting velocity. If we assume that no external forces affect the mass while in flight, then the trajectory of the point mass can be described as follows:

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

where $\mathbf{a}$ and $\mathbf{b}$ are the two parameters determined from the constraints of a ballistic motion, such as an origin and destination as well as traveling time constraint. M is the mass of a character and $\mathbf{g}$ is the gravitational constant. Ballistic paths are created in real time as animators move around the two end points and modify duration time. Fig. 3 shows the screen capture of the ballistic paths being manipulated. Multiple curves can be shown by varying the duration of the ballistic phase.

The animator can generate a physically-plausible path by setting a start and end point as constraints in 3D space.

Since the problem of generating a ballistic path between two points is under-constrained, the animator can also generate multiple ballistic trajectories, given a minimum and maximum time span. Thus each curve would present a proper ballistic trajectory, indicating the path an object would need to follow in order to meet the location constraints at differing times. Animators can also create ballistic paths by specifying the position and velocity of the start or the final frame.

Using the ballistic paths, one can easily correct linear momentum (hence the COM trajectory) of a ballistic motion of a character that an animator created; An animator chooses a desired ballistic path for a character, and then the system computes the character's COM at each frame and translate its root node so that the COM coincides with the current position of the ballistic path. The resulting motion is physically correct in terms of the linear momentum. Note that there are two ways to constrain the motion of a character or object to a ballistic path: an animator can either retain the start and end locations and adjust the timing to accommodate the ballistic path (Fig. 4), or they can manually adjust the ballistic path until it mostly matches the original path desired and then alter the motion of the character by retiming all the motion that occurs during the ballistic phase (Fig. 5). When adopting a ballistic path to the original motion path, the animator visually modifies the ballistic path until a good visual match is found.

We do not pay particular attention to the smoothness of the start and end of the ballistic trajectory. Rather, we use animator's original preparatory motion during the pre-ballistic and post-ballistic phases. We noticed that this does not introduce visual artifacts as long as the original motion is not very different than the corrected motion, which is usually the case. In the case where the artifacts are visible, animators can easily modify the frames in the pre- and post-ballistic phases during takeoff and landing using conventional animation tools.

Angular Momentum Tool

While there are many ways to change the angular momentum of a character, we develop a tool to change it by rotating only the global orientation of the character while keeping the original key frame animation of each body parts unchanged. This way, we 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. One important advantage of this algorithm is that modifying angular momentum is done online, giving prompt visual feedback to the animators.

For notational convenience, we will use the generalized notations for the velocity, inertia, and momentum, that provides

the combined representation of the angular and linear properties. We refer the readers to Appendix for the mathematical preliminaries of the generalized notations derived from Lie group theory. Additional details can be found in [24].

Let $\mathbf{v}_i$, $\mathbf{J}_i$, and $\mathbf{h}_i$ denote the generalized velocity, inertia, and momentum of a body part i , respectively, with the body part 0 being the root of the character. As shown in Appendix, a generalized velocity $\mathbf{v}_i$ of a body part i can be expressed as the sum of the velocity of the root and the relative velocity of i to the root:

$$\mathbf{v}_i = {}^i\mathbf{v}_0 + \mathbf{u}_i, \quad (2)$$

where ${}^i\mathbf{v}_0$ is the velocity of the root expressed in i 's body frame.² Likewise, we can divide the generalized momentum into two parts:

$$\mathbf{h}_i = \mathbf{J}_i \mathbf{v}_i \quad (3)$$

$$= \mathbf{J}_i ({}^i\mathbf{v}_0 + \mathbf{u}_i) \quad (4)$$

$$= \mathbf{J}_i {}^i\mathbf{v}_0 + \mathbf{a}_i, \quad (5)$$

where $\mathbf{a}_i = \mathbf{J}_i \mathbf{u}_i$ represents the momentum induced by the velocity of i relative to the root. Then the generalized momentum of a character in the root frame is computed as follows:

$${}^0\mathbf{h} = \sum_i {}^0\mathbf{h}_i \quad (6)$$

$$= \sum_i {}^0\mathbf{J}_i \mathbf{v}_0 + {}^0\mathbf{a}_i \quad (7)$$

$$= \left(\sum_i {}^0\mathbf{J}_i \right) \mathbf{v}_0 + \sum_i {}^0\mathbf{a}_i \quad (8)$$

$$:= {}^0\hat{\mathbf{J}} \mathbf{v}_0 + {}^0\mathbf{a}, \quad (9)$$

where $\hat{\mathbf{J}}$ is the composite rigid body inertia, an inertia if the multibody system of a current configuration were a single rigid body (i.e., all the joints of the character were locked). Note that ${}^0\hat{\mathbf{J}}$ and ${}^0\mathbf{a}$ are a function of joint angles only and independent of the motion of the root. Therefore, they remain constant while we manipulate the translation and rotation of the root. Using (9), we can efficiently compute the momentum of a character just by calculating $\mathbf{v}_0$ instead of recalculating velocities of every body part as we manipulate the root.

Our goal is to determine the configuration of the root $\mathbf{T}_0$ at a point in time such that the character has the user-specified momentum ${}^c\mathbf{h}^*$. We assume that the velocity of the character is computed from the configurations of the current and the previous time step, and that we will modify only $\mathbf{T}_0$ at the

current time step and keep $\mathbf{T}_0$ at the previous time step fixed. Since we keep the linear momentum fixed, the COM frame³ does not change while we rotate the character around the COM. Hence, given ${}^c\mathbf{h}^*$, its transformation with respect to the world frame ${}^w\mathbf{h}^* = \text{Ad}_{\mathbf{C}^{-1}}^* {}^c\mathbf{h}^*$ is also constant while we manipulate the character. For convenience, we will find $\mathbf{T}_0$ such that it creates ${}^w\mathbf{h}^*$.

The current momentum of the character with respect to the world frame is

$${}^w\mathbf{h} = \text{Ad}_{\mathbf{T}_0^{-1}}^* {}^0\mathbf{h}. \quad (10)$$

Note that both $\text{Ad}_{\mathbf{T}_0^{-1}}^*$ and ${}^0\mathbf{h}$ are functions of $\mathbf{T}_0$ and there does not exist a closed form solution. Therefore, our approach is to iteratively update $\mathbf{T}_0$ so that ${}^w\mathbf{h}$ approaches ${}^w\mathbf{h}^*$. Specifically, we will update $\mathbf{T}_0$ by $\mathbf{x}$ defined as

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

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

$$\delta\hat{\mathbf{v}}_0 = -(\mathbf{T}_0^{-1} \delta\mathbf{T}_0 \mathbf{T}_0^{-1}) \dot{\mathbf{T}}_0 + \mathbf{T}_0^{-1} \delta \dot{\mathbf{T}}_0 \quad (11)$$

$$= -\hat{\mathbf{x}} \hat{\mathbf{v}}_0 + \mathbf{T}_0^{-1} \frac{d}{dt} (\mathbf{T}_0 \hat{\mathbf{x}}) \quad (12)$$

$$= -\hat{\mathbf{x}} \hat{\mathbf{v}}_0 + \mathbf{T}_0^{-1} (\dot{\mathbf{T}}_0 \hat{\mathbf{x}} + \mathbf{T}_0 \dot{\hat{\mathbf{x}}}) \quad (13)$$

$$= -\hat{\mathbf{x}} \hat{\mathbf{v}}_0 + \hat{\mathbf{v}}_0 \hat{\mathbf{x}} + \dot{\hat{\mathbf{x}}} \quad (14)$$

$$\approx \text{ad}_{\hat{\mathbf{v}}_0} \hat{\mathbf{x}} + \frac{\dot{\hat{\mathbf{x}}}}{h} \quad (15)$$

where h is the time step. Using (15) and recalling that ${}^0\hat{\mathbf{J}}$ and ${}^0\mathbf{a}$ are constant, we can express $\delta^0\mathbf{h}$ in terms of $\mathbf{x}$:

$$\delta^0\mathbf{h} = {}^0\hat{\mathbf{J}} \delta\mathbf{v} \quad (16)$$

$$\approx {}^0\hat{\mathbf{J}} \left(\text{ad}_{\hat{\mathbf{v}}_0} \mathbf{x} + \frac{\mathbf{x}}{h} \right). \quad (17)$$

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

$$\delta^w\mathbf{h} = \delta(\text{Ad}_{\mathbf{T}_0^{-1}}^*) {}^0\mathbf{h} + \text{Ad}_{\mathbf{T}_0^{-1}}^* \delta^0\mathbf{h} \quad (18)$$

$$= -\text{Ad}_{\mathbf{T}_0^{-1}}^* \text{ad}_{\mathbf{x}}^* {}^0\mathbf{h} + \text{Ad}_{\mathbf{T}_0^{-1}}^* \delta^0\mathbf{h} \quad (19)$$

$$= \text{Ad}_{\mathbf{T}_0^{-1}}^* \text{ad}_{\hat{\mathbf{v}}_0}^* \mathbf{x} + \text{Ad}_{\mathbf{T}_0^{-1}}^* \delta^0\mathbf{h} \quad (20)$$

$$\approx \text{Ad}_{\mathbf{T}_0^{-1}}^* \left(\text{ad}_{\hat{\mathbf{v}}_0}^* + {}^0\hat{\mathbf{J}} \text{ad}_{\hat{\mathbf{v}}_0} + \frac{1}{h} {}^0\hat{\mathbf{J}} \right) \mathbf{x}, \quad (21)$$

where $\text{ad}_{\hat{\mathbf{v}}_0}^* \mathbf{x}$ and ${}^0\hat{\mathbf{J}} \text{ad}_{\hat{\mathbf{v}}_0} \mathbf{x}$ account for the effect of coordinate change of $\mathbf{T}_0$ due to $\mathbf{x}$ on ${}^0\mathbf{h}$ and $\mathbf{v}_0$, respectively, and $\frac{1}{h} {}^0\hat{\mathbf{J}} \mathbf{x}$ is the added momentum due to $\mathbf{x}$. By solving (21) for $\mathbf{x}$, we can compute $\mathbf{x}$ that will create desired $\delta^w\mathbf{h}$.

Based on the relations derived above, the algorithm to compute $\mathbf{T}_0$ given ${}^c\mathbf{h}^*$ is as follows:

³The COM frame is defined as a coordinate frame that is parallel to the world frame with its origin coinciding with the COM.

²Left superscript indicates the reference frame of a symbol. ${}^0\mathbf{v}_i$, ${}^c\mathbf{v}_i$, and, ${}^w\mathbf{v}_i$ are $\mathbf{v}_i$ expressed in the root frame, COM frame, and the world frame, respectively. We do not use the left superscript when the symbol is expressed in its own body frame. For example, $\mathbf{v}_i = {}^i\mathbf{v}_i$.

- 1: ${}^w\mathbf{h}^* = \text{Ad}_{\mathbf{C}^{-1}}^* {}^c\mathbf{h}^*$, where $\mathbf{C}$ is the COM frame with respect to the world frame.
- 2: Compute ${}^0\hat{\mathbf{J}}$ and ${}^0\mathbf{a}$.
- 3: Compute ${}^w\mathbf{h}$
- 4: **while** $\delta {}^w\mathbf{h} = {}^w\mathbf{h}^* - {}^w\mathbf{h}$ is above a threshold **do**
- 5: Compute $\mathbf{x}$ by solving (21).
- 6: $\mathbf{T}_0 \leftarrow \mathbf{T}_0 e^{\gamma \hat{\mathbf{x}}}$ where $0 < \gamma \leq 1$.
- 7: Update $\mathbf{v}_0$, ${}^0\mathbf{h}$ (Eq. (9)), and ${}^w\mathbf{h}$ (Eq. (10)).

γ controls the distance for the next iteration. Note that we will change only the angular part of ${}^c\mathbf{h}^*$ and keep the linear momentum fixed. However, both angular and linear part of ${}^w\mathbf{h}^*$ will be changed due to the coordinate transformation.

PHYSICAL ACCURACY OF KEY FRAME ANIMATIONS

In this section, we investigate the physical accuracy of the keyframe animations created by professional animators. Specifically, we compute the center of mass, linear momentum, and angular momentum of the ballistic motions and investigate how accurately these properties follow Newton's laws. For walking and running animations, we compute the COP and verify that it is actually within the support polygon.

Ballistic Motions

We collect 13 ballistic motions of human-like characters such as jumping and falling motions. The durations of each motion ranges from 4 to 27 frames. In total, 150 frames are examined.

In ballistic phase, the path of COM projected to the horizontal plane should form a straight line. Fig. 6 (a) is the histogram of errors in the horizontal plane of the ballistic motions. Error is defined as the distance from the horizontal projection of COM to the line segment connecting the horizontal projection of COM's of the first and the last frames. The errors are normalized by the length of the line segment. The figure shows that the errors are less than 0.1 for more than 90% of frames. This relatively low error seems to be due to the way that animators create key frames for ballistic motions. Since animators usually create a straight line from the start and end position of the root node and adjust the heights of the in-between key frames, the horizontal projection of COM trajectory shows a more or less straight line unless a character changes its pose a lot.

Fig. 6 (b) and (c) show the histogram of normalized errors of the linear momentum in horizontal (X-Z) plane. Error is defined as the difference between the current momentum and the average momentum, normalized by the magnitude of the average momentum. Comparing the error of linear momentum with that of COM, we can see that, even though in-between

key frames make a more or less straight line, the location of the key frame in the line is less correct from the perspective of the linear momentum.

Fig. 6 (d), (e), and (f) show the histogram of normalized errors of the angular momentum in X, Y, and Z directions, respectively. As in the linear momentum, error is defined as the difference between the current momentum and the average momentum. We can see that the error in the angular momentum is much greater than that of the linear momentum. We conjecture that one of the reasons is that the angular momentum is less perceptible than linear momentum to human eyes. Except for flipping jumps where the rotation of the whole body is explicit, angular momentum of the ballistic motion is actually hard to perceive without a visualization tool such as the one in this paper.

By fitting Eq. 1 to the COM trajectory, we estimate the gravity of a scene. Table I shows such gravity relative to earth gravity (9.8 m/s^2) for each scene. The table shows a great deviation of gravity among scenes ranging from 0.5g to 3.3g. Notably, the median of the gravities is 0.95g, quite close to the real gravity. This observation shows that the key framed animations create ballistic motions under true gravity on average, but there are large deviations among scenes. Research has shown that deviations in horizontal or vertical accelerations can be detected by an observer [25], and our data demonstrates that hand-animated motion often exceeds such thresholds. Another interpretation of the variation in estimated gravity is that there is a large variance in animators' notion of a character's size. Lower gravity in the data indicates that the animator considers the character bigger than it actually is, and higher gravity means the animator thinks that the character is smaller than it is. Refer to the Discussion section for the effect of a character's scale on the speed of ballistic motions. Other research has suggested that preparatory motion may also be an important consideration in the perceived visual quality of ballistic motion [26]. Our method does not address the preparatory or recovery phases of ballistic motion, and leaves such a task to the animator.

Ambulatory Motions

We collected 5 scenes of walking motions (165 frames) and 5 scenes of running motions (113 frames) of biped characters, and examined whether the COP lies inside the support polygon. Fig. 7 shows the histogram of normalized distance of the COP from the support polygon. The distance is positive when the COP is outside the support polygon and it is normalized by the mean radius of the support polygon, computed as the perimeter divided by 2π .

For walking motions, the COP falls inside the support polygon in more than 60% of frames, and it is within the normalized distance of 1 in 85% 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 for the realism of ambulatory motions. Somewhat naturally, the error grows as the speed of a character increases. For running motions, the COP is within the support polygon in only 33% of frames, and it is within the normalized distance of 1 in 70% of frames.

DISCUSSION

Effectiveness and Impact

Many animators found this system useful for improving physical realism of the keyframed animation. Note that the manually created animation in Fig. 4 and in the accompanying video is the result of the substantial amount of time and effort devoted by professional animators. As a result, some examples in the accompanying video are already quite realistic, leaving less room for the improvement using our tool. In some sense, this shows that physical realism is one of the key factors for high quality animation. Other examples show that our system improves the physical realism significantly. Also note that when used from the beginning of the animation process, our system may increase the efficiency of the animation making.

Notably, we found that the visualization tools introduced in this paper can also serve as a gavel. Individual animators may have their own sense of physical correctness, which can cause debate over how a character should move when collaborating animators have a different sense of physical intuition. We have observed that our tool helps animators reach an agreement on physically correct animation by quantifying the discrepancy in the motion. For example, the ballistic path can indicate exactly how many frames 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 utilizing animated characters have ballistic motion such as falling or jumping, of which many could be improved through the use of this system. Of course, many more shots benefit from COM and COP visualization.

Our system is most effective when the animator understands how the physical curves should be changed in order to create better animations. Thus, it was useful to create videos that showed, for example, the proper location of the COM and COP (or ZMP) during a walk, run and so forth. Such training videos are effective at teaching the animator how to interpret

our system, as in Fig. 8. Of course, whether or not the animator wants their character to act human-like is a choice left to them.

Use of COM and COP

Experienced and skilled animators intuitively know how to make animations look better than inexperienced and lesser skilled animators. We noticed that the COM and COP guidelines explicitly expressed the notions of physical correctness in animations. For example, the COM of a humanoid will tend 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, since, for example, the idea of “carrying the weight from one foot to another” could be directly seen in the projection of the COM.

Scaling Large and Small Characters

The scale of the animated characters can dramatically impact the resulting animation. Other research has provided a detailed explanations for scaling dynamic control systems based on time, positions and velocities [27]. 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, as small creatures likewise appear to move more quickly. The time t required for an object to travel to a maximum height h is given by:

$$t = \sqrt{2h/g}, \quad (22)$$

where g is the force of gravity. Thus, movements related to gravity, such as walking where gravity pulls the character's swing leg to the ground, or throwing objects from the character, 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 are 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 (Fig. 9).

Limitations and Future Work

Our method to modify angular momentum only changes the global orientation of the character while preserving 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 their angular momentum by windmilling their arms in a circular manner. We do not provide a tool to let animators explore all these possibilities, instead focussing on ease-of-use and automation. It is not clear that there exists a single useful correction method which will yield better animation due to animation constraints, such as requiring a character's feet to be in contact with the ground during landing. Thus, it is the animator's role to manually correct the posture of the character to resolve this discrepancy. In the above example, the angular momentum during flight can be automatically adjusted, but would need to be manually smoothed with the landing posture to obtain correctness.

In this paper, the COP is computed under the assumption that the ground is flat. However, it is common that characters walk or run on uneven ground. When the contact points between the character and the ground are not coplanar, the support polygon cannot be defined. In this case, we will need to extend the notion of the support polygon to 3D space.

CONCLUSION

We have presented a system that allows an animator to view the dynamic aspects of the character animation that they have created through kinematic means and modify the original animation in order to make the appearance of the motion more physically feasible. The strength of our approach is that the algorithms used to generate physical visualization are straightforward to implement, and compatible with most existing kinematic animation systems. In addition, the system does not require that an animator change the methods by which she generates animation, thus leveraging existing animator skill. In addition, the quality of the motion generated is additive - it does not replace the underlying animation, and is only used if it enhances the realistic appearance of the motion, all fully at the decision and control of the animator.

Although our system was designed for live-action visual effects, the techniques can be used for almost any animation purpose that requires or desires better physical realism in character animation, such as systems for fully 3D environments, prebaked animations for video games, and so forth.

Our system can also help to enforce consistency of character motion across an entire animation studio. It is typical for multiple animators to each design their own animations for a single animated character in different scenes. In fact, a large visual effects show might require the coordination of dozens of animators in order to produce hundreds of animations for a small number of characters. Animations produced by

different animators using the same character will tend to be more consistent with each other while using our system, since animators do not have to solely rely on their individual senses of timing and space, but rather can use the interactive visual feedback that our system provides.

APPENDIX

Given a homogeneous representation of a moving body frame $\mathbf{T}(t) = (\mathbf{R}, \mathbf{p})$, where $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ 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

$$\hat{\mathbf{v}} = \mathbf{T}^{-1} \dot{\mathbf{T}} = \begin{bmatrix} [\omega] & v \\ \mathbf{0} & 0 \end{bmatrix}, \quad (23)$$

where 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 , i.e., $[\omega]\eta = \omega \times \eta$ for all $\eta \in \mathbb{R}^3$. We use $\hat{\mathbf{v}}$ for the 4-by-4 matrix representation of $\mathbf{v}$. The generalized momentum $\mathbf{h} = [\mathbf{k}^T, \mathbf{l}^T]^T$ is defined as

$$\mathbf{h} = \mathbf{J}\mathbf{v}, \quad (24)$$

where $\mathbf{k} \in \mathbb{R}^3$ and $\mathbf{l} \in \mathbb{R}^3$ represent the angular and the linear momentum (with respect to the body frame), respectively, and the *generalized inertia* $\mathbf{J} \in \mathbb{R}^{6 \times 6}$ of the rigid body has the following structure:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & m[\mathbf{r}] \\ m[\mathbf{r}]^T & m\mathbf{1} \end{bmatrix}, \quad (25)$$

where m is the mass, $\mathbf{I} \in \mathbb{R}^{3 \times 3}$ is the rotational inertia matrix, $\mathbf{r} \in \mathbb{R}^3$ is the position of the center of mass, and $\mathbf{1} \in \mathbb{R}^{3 \times 3}$ is the identity matrix. Eq. (24) is coordinate-invariant; i.e., 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 $\text{Ad}_{\mathbf{T}}$ is defined as $\text{Ad}_{\mathbf{T}} \hat{\mathbf{g}} = \mathbf{T}\hat{\mathbf{g}}\mathbf{T}^{-1}$, or in matrix form as

$$\text{Ad}_{\mathbf{T}} \mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ [\mathbf{p}]\mathbf{R} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix}. \quad (26)$$

The adjoint mapping is used in the coordinate transformation of the generalized velocity. The corresponding dual adjoint mapping that performs the coordinate transformation of the generalized momentum is $\text{Ad}_{\mathbf{T}}^*$ and it is defined as the transpose of $\text{Ad}_{\mathbf{T}}$; i.e., $\text{Ad}_{\mathbf{T}}^* = \text{Ad}_{\mathbf{T}}^T$. For example, the generalized velocity, momentum, and inertia with respect to the world frame are

$${}^w\mathbf{v} = \text{Ad}_{\mathbf{T}}\mathbf{v} \quad (27)$$

$${}^w\mathbf{h} = \text{Ad}_{\mathbf{T}^{-1}}^*\mathbf{h} \quad (28)$$

$${}^w\mathbf{J} = \text{Ad}_{\mathbf{T}^{-1}}^*\mathbf{J}\text{Ad}_{\mathbf{T}^{-1}}. \quad (29)$$

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

TABLE I: Gravity in test scenes. g: measured gravity/9.81.

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

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

$$\mathbf{T}_i = \mathbf{T}_0 \mathbf{G}_i, \quad (30)$$

where $\mathbf{T}_0$ is the configuration of the root and $\mathbf{G}_i$ denotes the relative configuration of $\{i\}$ with respect to the root. Substituting (30) into (23), we can decompose the body velocity $\mathbf{v}_i$ as follows:

$$\mathbf{v}_i = {}^i\mathbf{v}_0 + \mathbf{u}_i \quad (31)$$

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

$$\begin{bmatrix} [\omega_u] & v_u \\ \mathbf{0} & 0 \end{bmatrix} = \mathbf{G}_i^{-1} \dot{\mathbf{G}}_i. \quad (32)$$

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

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

$$\text{ad}_{\mathbf{g}_1} \mathbf{g}_2 = \begin{bmatrix} [\omega_1] & 0 \\ [v_1] & [\omega_1] \end{bmatrix} \begin{bmatrix} \omega_2 \\ v_2 \end{bmatrix}. \quad (33)$$

Its dual $\text{ad}_{\mathbf{g}}^*$ for the generalized momentum is defined as its transpose $\text{ad}_{\mathbf{g}}^* = \text{ad}_{\mathbf{g}}^T$. Note that $\text{ad}_{\mathbf{g}} \mathbf{v} = -\text{ad}_{\mathbf{v}} \mathbf{g}$ and $\text{ad}_{\mathbf{g}} + \text{ad}_{\mathbf{v}} = \text{ad}_{\mathbf{g}+\mathbf{v}}$. The Lie bracket occurs when Ad is differentiated. For example,

$$\frac{d}{dt} \text{Ad}_{\mathbf{T}} = \text{Ad}_{\mathbf{T}} \text{ad}_{\mathbf{v}} \quad (34)$$

where $\mathbf{v}$ is the body velocity of $\mathbf{T}$. Refer to [28] for the proof.

REFERENCES

- [1] Z. Kačić-Alesić, M. Nordenstam, and D. Bullock, "A practical dynamics system," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*. Eurographics Association, 2003, pp. 7–16.
- [2] S. Regelous, *Crowd and Group Animation*. SIGGRAPH Course Notes, 2005.

Fig. 1: Center of mass projected on the ground (red circle), support polygon (red polygon) and the center of pressure (purple circle) for key framed animation. Note that the horizontal shape of the character (right) causes a greater amount of instability in the COP calculation than humanoid character (left).

Fig. 2: Visualization of angular momentum, shown as a vector protruding from the center of mass of the character. The yellow arrows indicate the direction of the motion (using the right-hand rule), while the size of the vectors indicate the relative amount of rotation about that axis.

Fig. 3: (Left) Tool to generate multiple ballistic paths between two locators. (Right) This can be used to pregenerate paths between two points in a scene such as jumping or falling off a tall structure.

Fig. 4: The blue curve is the trajectory of the center of mass of the character's animation that an animator created manually. Our system suggests the physically correct ballistic path (the red curve) that the character's center of mass should follow. The system allows an animator to automatically change the original animation to match physical laws.

Fig. 5: Character walking and jumping. Notice that how the ballistic path (red) requires two additional frames to complete the trajectory than does the animated path (blue). Also note that the keys for the ballistic path, which determine the timing, appear at slightly different locations than those in the animated path.

- [3] Havok Inc., "Havok behavior," <http://www.havok.com/>, 2007.
- [4] NaturalMotion Ltd., "Endorphin," <http://www.naturalmotion.com/>, 2007.
- [5] A. Witkin and M. Kass, "Spacetime constraints," in *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1988, pp. 159–168.
- [6] C. K. Liu, A. Hertzmann, and Z. Popović, "Learning physics-based motion style with nonlinear inverse optimization," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1071–1081, 2005.
- [7] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien, "Animating human athletics," in *Computer Graphics (SIGGRAPH '95 Proceedings)*, 1995, pp. 71–78.
- [8] P.-F. Yang, J. Laszlo, and K. Singh, "Layered dynamic control for interactive character swimming," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004, pp.

Fig. 6: Measurement of physical correctness of ballistic motions created by professional animators using conventional key framing process. (a) Normalized error of COM in horizontal plane. (b),(c) Normalized error of linear momentum in X and Z directions, respectively. (d)-(f) Normalized error of angular momentum in X, Y, and Z directions, respectively.

Fig. 7: Histogram of normalized distance between the COP and the support polygon for walking (left) and running (right) created by professional animators using conventional key framing process.

39–47.

- [9] J. Laszlo, M. van de Panne, and E. Fiume, "Limit cycle control and its application to the animation of balancing and walking," in *Computer Graphics (SIGGRAPH '96 Proceedings)*, 1996, pp. 155–162.
- [10] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *Proceedings of ACM SIGGRAPH 2001*, ser. Computer Graphics Proceedings, Annual Conference Series, Aug. 2001, pp. 251–260.
- [11] V. B. Zordan, B. Celly, B. Chiu, and P. C. DiLorenzo, "Breathe easy: Model and control of simulated respiration for animation," in *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004, pp. 29–37.
- [12] N. S. Pollard and V. B. Zordan, "Physically based grasping control from example," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005, pp. 311–318.
- [13] A. Shapiro, F. H. Pighin, and P. Faloutsos, "Hybrid control for interactive character animation," in *11th Pacific Conference on Computer Graphics and Applications*, 2003, pp. 455–461.
- [14] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast, "Dynamic response for motion capture animation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 697–701, 2005.
- [15] M. Neff and E. Fiume, "Modeling tension and relaxation for computer animation," in *SCA '02: Proceedings of the 2002 ACM SIG-*

Fig. 8: Using motion capture data as a training tool for animators. Notice that the position of the COM is outside the support polygon while the character is turning.

Fig. 9: Example of how scale affects the appearance of motion. Shown are a normal-sized man and one 10 times the size. A ball is placed in motion around both men. The ball moving around the larger man will take 3.19 times as long as it will around the normal-sized man.

GRAPH/Eurographics symposium on Computer animation, 2002, pp. 81–88.

- [16] —, “Methods for exploring expressive stance,” in *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004, pp. 49–58.
- [17] A. Majkowska and P. Faloutsos, “Flipping with physics: motion editing for acrobatics,” in *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 35–44.
- [18] S. Tak and H.-S. Ko, “A physically-based motion retargeting filter,” *ACM Trans. Graph.*, vol. 24, no. 1, pp. 98–117, 2005.
- [19] H. J. Shin, L. Kovar, and M. Gleicher, “Physical touch-up of human motions,” in *11th Pacific Conference on Computer Graphics and Applications*, Oct. 2003, pp. 194–203.
- [20] M. B. Popovic, A. Goswami, and H. Herr, “Ground reference points in legged locomotion: Definitions, biological trajectories and control implications,” *International Journal of Robotics Research*, vol. 24, no. 12, pp. 1013–1032, 2005.
- [21] A. Sulejmanpasić and J. Popović, “Adaptation of performed ballistic motion,” *ACM Transactions on Graphics*, vol. 24, no. 1, pp. 165–179, Jan. 2005.
- [22] P. S. A. Reitsma and N. S. Pollard, “Perceptual metrics for character animation: sensitivity to errors in ballistic motion,” in *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM, 2003, pp.

537–542.

- [23] R. McDonnell, F. Newell, and C. O’Sullivan, “Smooth movers: perceptually guided human motion simulation,” in *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 259–269.
- [24] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. New York: CRC Press, 1994.
- [25] P. S. A. Reitsma and N. S. Pollard, “Perceptual metrics for character animation: sensitivity to errors in ballistic motion,” in *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM, 2003, pp. 537–542.
- [26] 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,” *Comput. Graph. Forum*, vol. 27, no. 2, pp. 201–210, 2008.
- [27] N. Pollard, “Simple machines for scaling human motion,” in *In Proceedings of Computer Animation and Simulation '99*, August 1999.
- [28] S.-H. Lee, J. Kim, F. Park, M. Kim, and J. Bobrow, “Newton-type algorithms for dynamics-based robot movement optimization,” *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 657–667, 2005.

Ari Shapiro Ari Shapiro currently works as a Graphics Scientist at Rhythm & Hues Studios. He has worked for three years in the visual effects industry and over 16 years in the software field. His research interests include character motion and physics-based animation. He received his BA in Computer Science from UCSC in 1999, an MS in Computer Science from UCLA in 2002, and a PhD in Computer Science from UCLA in 2007.

Sung-Hee Lee Sung-Hee Lee is a Postdoctoral Researcher at Honda Research Institute, USA. His research interests include human modeling and animation, physics-based animation, multibody dynamics, and humanoid robotics. Lee received his BS and MS in mechanical engineering from Seoul National University in 1996 and in 1998, and PhD in computer science from University of California, Los Angeles in 2008. He is a member of ACM SIGGRAPH.