Acceleration Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters

Niranjan Kalyanasundaram
nkalyan@clemson.edu

Acceleration Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters

Niranjan Kalyanasundaram

Acceleration Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Digital Production Arts

---

by
Niranjan Kalyanasundaram
December 2021

Abstract

Secondary effects are key to adding fluidity and style to animation. This thesis introduces the idea of "Acceleration Skinning" following a recent well-received technique, Velocity Skinning, to automatically create secondary motion in character animation by modifying the standard pipeline for skeletal rig skinning. These effects, which animators may refer to as *squash and stretch* or *drag*, attempt to create an illusion of inertia. In this thesis, I extend the Velocity Skinning technique to include acceleration for creating a wider gamut of cartoon effects. I explore three new deformers that make use of this *Acceleration Skinning* framework: *followthrough*, *centripetal stretch*, and *centripetal lift* deformers. The followthrough deformer aims at recreating this classic effect defined in the fundamental principles of animation. The centripetal stretch and centripetal lift deformers use rotational motion to create radial stretching and lifting effects, as the names suggest. I explore the use of effect-specific time filtering when combining these various deformations together, allowing for more stylized and aesthetic results. I finally conclude with a production evaluation, exploring possible ways in which these techniques can be used to enhance the work of an animator without losing the essence of their art.

Dedication

I would like to dedicate this thesis to my parents for supporting me wholeheartedly in my decision to let go of a stable career to move to America and pursue a masters degree. I would not be where I am today without their love and guidance.

I would also like to extend this dedication to the memories of the late professor Sundara Pandiyan for inspiring me to pursue a higher education in this field of study, and the late Harrison Diesl for being a true friend and an unflinching source of support to all of us in the Digital Production Arts program at Clemson.

Acknowledgments

I acknowledge and thank the artists, Rodney Costa, Mohammad Saffar, and Kayla Rutherford for volunteering to be a part of our usability study. Specifically, I thank Rodney Costa for contributing character rigs, and assisting in finding workarounds to pipeline issues faced when extracting data from Maya to our application. I thank Mohammad Saffar and Kayla Rutherford for animating these rigged characters, thereby producing high quality input for our application, and allowing us to experiment and attempt enhancing the animation through our application. Their help and feedback was integral to the completion of this project. I thank Reagan Burke whose insightful question prompted discussions for future extensions to the work presented here.

I would also like to thank Dr. Victor Zordan and Dr. Damien Rohmer for their patience and continued guidance in this research. I thank Dr. Eric Patterson and Dr. Jerry Tessendorf for consenting to be a part of my review committee, and for their invaluable comments and feedback. I thank all of my professors at Clemson University for molding me into the person that I am today.

Finally, I would like to thank all of my classmates and fellow students at Clemson for sharing in the trauma commonly known as graduate school and being a part of this wonderful journey.

Thank you.

Table of Contents

List of Figures

Chapter 1

Introduction

The twelve fundamental principles of animation laid out by The Nine Old Men of Walt Disney Animation Studios have been a driving factor for animators everywhere [33]. These principles, created by the pioneers of traditional hand-drawn animation, have been guiding artists on how to create believable and lively characters for almost a century. Principles such as *squash and stretch* and *followthrough* add vitality to animation. Since the dawn of computer animation, many new techniques were created to adapt these principles to suit the world of 3D animation [21]. However, this is quite a challenging task. Traditional animators were primarily restricted by only one limitation: their skills as an artist. An exceptional traditional animator can put pencil to paper, draw shapes and figures moving the way they desire, and bring characters to life. 3D animation, on the other hand, is not quite as straightforward. The animator must be provided with a rigged and skinned character with which to animate. Much like a puppeteer is limited by the capabilities of their marionette, 3D animators are limited by the deformations supported by their character rig.

Rigging is the process of creating a skeletal structure for a character to allow an animator to control it. Skinning is process of "binding" the character mesh to these skeletal joints so that when the joints move, the corresponding parts of the character mesh are deformed along with it. Standard approaches to skinning a character include methods such as Linear Blend Skinning (LBS) or Dual Quaternion Skinning (DQS) (see section 2.3 for further details). Many principles of animation, such as squash and stretch, also require the characters shape to be aesthetically morphed based on the underlying motion of the character. The common approach to achieving such effects in 3D is to apply non-linear deformers onto the character mesh; these deformers transform the target mesh by

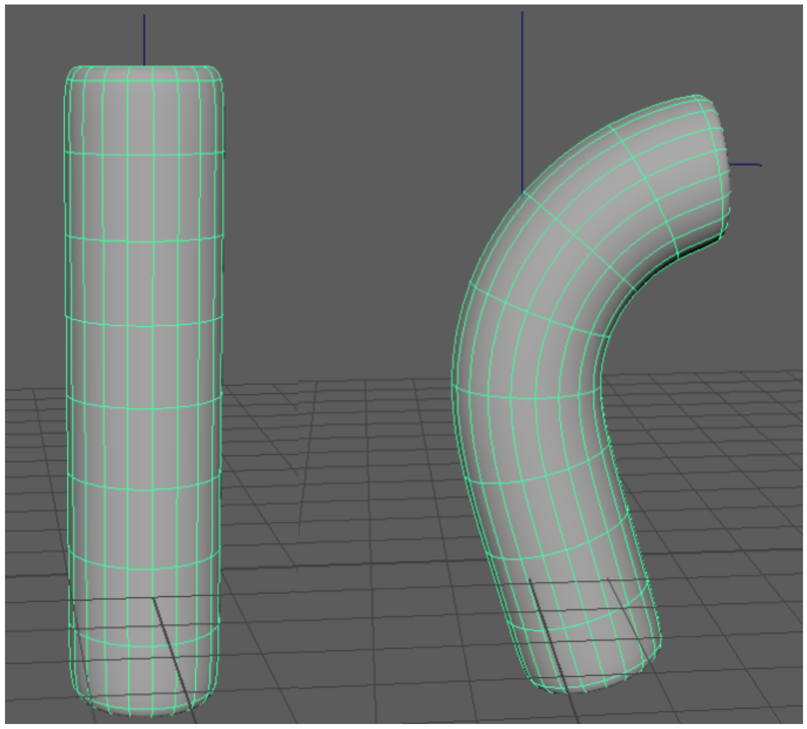

Figure 1.1: Non-linear bend deformer applied onto a cylinder in Autodesk Maya

translating vertices based on certain user-driven parameters. This requires animators to manually tweak these parameters to create the desired effects. For example, Fig. 1.1 showcases a non-linear bend deformer applied to a cylinder in Autodesk Maya. The user has applied curvature and adjusted the lower and upper bounds of the deformer independently.

Novel approaches to the rigging and skinning process have been proposed over the years; Many of these approaches attempt to enhance deformation to emulate these animation principles. These approaches can be broadly split into two categories:

- **Geometric Approaches**: These approaches primarily use shapes and geometry to guide their deformation. Such methods generally require less computation power, making them more interactive. This factor also typically makes these techniques more user-friendly. However, such approaches may place more of an onus on the artist to craft the deformations the way they desire, thereby making it less physically accurate. It is up to the artist to painstakingly create the effect they desire.

- **Physically-Based Simulations**: These approaches introduce physical forces and mass to compute simulations over time intervals. The results of these simulations are then typically used to deform the skinned mesh. While these methods provide more accurate physical deformations, they generally require more computation power, making them less interactive. While the simulation is itself generally automated (the user supplies predefined values and runs the simulation), such methods require a trial-and-error approach to achieve the required results, making them less directable.

1.1 Primary Contributions

The recently introduced “Velocity Skinning” technique is a geometric approach to skinning which uses the linear and angular velocities of skeletal joints to guide deformation of skinned mesh vertices [32]. This method accepts a skinned skeletal rig as input. The artist animates their character as usual. Velocity Skinning then further displaces each vertex on the mesh (in addition to standard skinning) using a set of deformation functions. These functions are guided by the kinematic motion of the skeletal joints. Through this method, effects such as squash and stretch or $floppy$ followthrough are inherently present as part of the rig and need not be manually configured during rigging. The deformations also expose parameters to allow enhanced artist control, if desired. Finally, Velocity Skinning is proposed as a generic framework; its mathematics enable the addition of new deformation functions with ease.

In this thesis, I provide an overview of the recently created “Velocity Skinning” framework and propose enhancements to this technique. The primary contributions of this thesis are as follows:

- I expand the Velocity Skinning technique by also considering the acceleration of skeletal joints, proposing a new system: Acceleration Skinning. This includes the utilization of the following acceleration terms:
  - Linear Acceleration a
  - Angular Acceleration α
  - Tangential Acceleration a_t
  - Centripetal Acceleration a_c
- I use these acceleration terms to propose three new deformation functions, further expanding the gamut of effects supported by this framework:

  - Followthrough
  - Centripetal Stretch
  - Centripetal Lift

- I propose the use of effect-specific time filtering to independently modify the reactivity of each deformer.

- I propose enhancements to the existing Velocity Skinning system to allow more refined artist control (these enhancements, namely twist-bend decomposition and local joint control, were included in the official Velocity Skinning publication [32]).

- Finally, I conduct a study to evaluate the usability of these tools in a practical production scenario.

The benefits of Acceleration Skinning are:

- Like Velocity Skinning, it remains a geometric approach, making it fast to compute.
- It is artist-driven and easy to control.
- It is a generic framework that supports any skeletal rig. It is not restricted to any particular skeletal hierarchy (such as biped or quadruped hierarchies).
- It expresses the supported deformation effects as part of the skinning process.
- The range of supported effects can easily be expanded. New deformation functions may be implemented to utilize these same kinematic terms to create new effects.
- Each deformer function provides a set of input parameters that an artist may tweak to tune the effect to their liking.

1.2 Terminology

We slightly deviate from the terminology used in traditional animation (defined by the fundamental principles of animation) for practical purposes. This deviation is best explained with an example. Consider a cloth draped over a character. When the character moves, the cloth *drags* behind them. When the character comes to a stop, the cloth overshoots the characters rest position before settling back down. In traditional animation, this entire process of dragging and settling is referred to as $followthrough$ [33] (see section 2.2 for more detailed concepts). Within our framework, Velocity Skinning provides a deformer to handle $floppy\ drag$ deformation (like the effect of the cloth dragging behind the character). However, this deformer does not allow the skinned mesh to overshoot the skeletal position. Acceleration skinning proposes a new deformer created specifically to model this remaining portion of the effect: overshooting and settling down. Both of these effects are decoupled and can be controlled independently. For convenience, in this thesis, we refer to the first effect as $drag$ and the second effect as $followthrough$. This naming convention is somewhat consistent with the terminology described in The Animator's Survival Kit [36] (see Fig. 2.1).

Chapter 2

Related Work

2.1 Historical Growth of Animation

Animation has existed as a medium of artistic expression for over a century. Looking back through history, one can see the art grow and develop from humble beginnings. We have record of cave wall paintings from over 35,000 years ago depicting animals with extra sets of legs, intended to portray motion [36]. The Zoetrope, a cylindrical toy from the 1860s, lets viewers peer through slits in a rotating cylinder to catch momentary glimpses of sequential drawings pasted on the inside [36]. Spinning the cylinder creates the illusion of movement. Cartoonist J. Stuart Blackton in his work "The Enchanted Drawing" films himself drawing a face, and cleverly uses film edits to change the expressions on the face to react to his actions [16]. One may argue that there is no actual *animation* in this short film, however the idea to amalgamate drawings and motion picture to convey emotion in this fashion was fresh and imaginative [24]. In 1914, Winsor McCay animated a cartoon dinosaur. In a live performance, he projected the animation onto a screen and "interacted" with it. This work, "Gertie the Dinosaur", is credited as the first historical example of character animation: an animated character with personality and individuality that audiences could relate to [36] [37]. In the 1920s, Walt Disney had the novel idea of synchronizing cartoons with sound. 1928 saw the release of Disney's famous "Steamboat Willie", the first ever released animated cartoon featuring Mickey Mouse [2]. Funneling further efforts into following up this success, Disney and his animators became adept in the art of creating sound cartoons [24]. This time period saw a marked rise in other animation studios such as Fleischer Studios by the brothers Max and Dave, who rose to fame with the creation of their character, $Ko - Ko\ the\ Clown$. Production companies began investing in animation as well, most notably Warner Bros. funding Leon Schlesinger Productions to create the Looney Tunes animated series [11] [4]. This competitive atmosphere drove the animation industry into what eventually became known as "The Golden Age of Animation" [11]. The invention of computers, followed closely by the ingenuity of engineers, heralded in the age of computer graphics and animation. In 1972, Ed Catmull created groundbreaking work in 3D computer graphics in his work "A Computer Animated Hand" [10] [14]. Advances in technology such as this paved the way for Pixar Animation Studios to create the world's first 3D animated feature length film [10]. From Gertie the Dinosaur to the modern, bleeding-edge renders of 3D animation, virtual reality, and visual effects, animation has certainly come a long way. However, the goals behind animation have mostly remained unchanged since the early days; a desire for visually conveying an artistic expression of character. While the technology involved in this expression has undoubtedly evolved through the ages, the foundational concepts behind such creative expression were developed by the early pioneers of animation based on their keen observation of physical movement.

2.1.1 Fundamental Principles of Animation

The fabled nine old men of Walt Disney Animation Studios are credited to have had a significant hand in consolidating the art form into a set of "twelve fundamental principles of animation" [33]. Certain principles, such as staging, arcs, solid drawing, and exaggeration, deal with getting the character silhouettes and movement to read easily on-screen, focusing on bringing the point across to the audience quicker. Other principles, including squash and stretch, anticipation, follow through and overlapping action, secondary action, and more involve enhancing the aesthetics of the motion itself. These principles attempt to make the character motion more believable and appealing to an audience. For example, stretch and squash is one of the most important of these twelve principles, as it is instrumental in conveying an illusion of gravity and other forces. For example, when a bouncing ball is in mid flight, it may stretch a bit in the direction of movement. When it collides with another surface, it squashes on impact before expanding again [8].

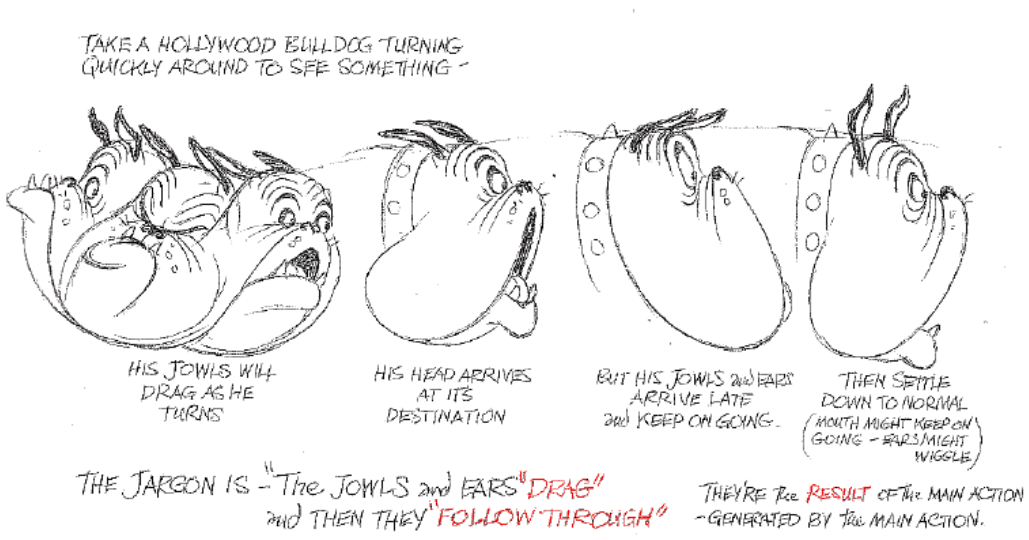
TAKE A HOLLYWOOD BULLDOG TURNING QUICKLY AROUND TO SEE SOMETHING -
HIS JOWLS WILL DRAG AS HE TURNS
HIS HEAD ARRIVES AT ITS DESTINATION
BUT HIS JOWLS and EARS ARRIVE LATE and KEEP ON GOING.
THEN SETTLE DOWN TO NORMAL
(MOUTH MIGHT KEEP ON GOING - EARS MIGHT WIGGLE)
THE JARGON IS - "The JOWLS and EARS "DRAG" and THEN THEY "FOLLOW THROUGH"
THEY'RE the RESULT of the MAIN ACTION - GENERATED BY the MAIN ACTION.

Figure 2.1: Drag and follow-through effect visible on the ears and jowls of a bulldog animation. Source: The Animator's Survival Kit [36]

2.2 Followthrough and Overlapping Action

Another essential principle is that of "follow through and overlapping action." This principle came to be at Walt Disney Studios due to Disney's disapproval of his animators' quality of work. In a review session, he is quoted to have said, "Things don't stop all at once, guys; first there's one part and then another." [36]. This ideology propelled and still effectively summarizes the creation of this principle of animation. We can observe in real life that the movement of organic objects or beings occurs in stages or parts. Loose appendages rarely move on their own; they are affixed, either naturally or intentionally, upon a primary source of action. When in motion, such appendages drag behind the primary action. When the primary source comes to rest, the appendages overshoot beyond the rest position before settling down. For example, one can think of loose hanging clothes, long ears or tails on a dog, or loose hanging flesh like a big belly. Animating such phenomena requires the use of followthrough to effectively abstract reality. In essence, this principle attempts to capture and express inertia artistically. Examples from seminal works in animation include the jowl animation on a Hollywood bulldog (Fig. 2.1) or the cloth and arm animation on this simple character (Fig. 2.2), both examples sourced from The Animator's Survival Kit [36].

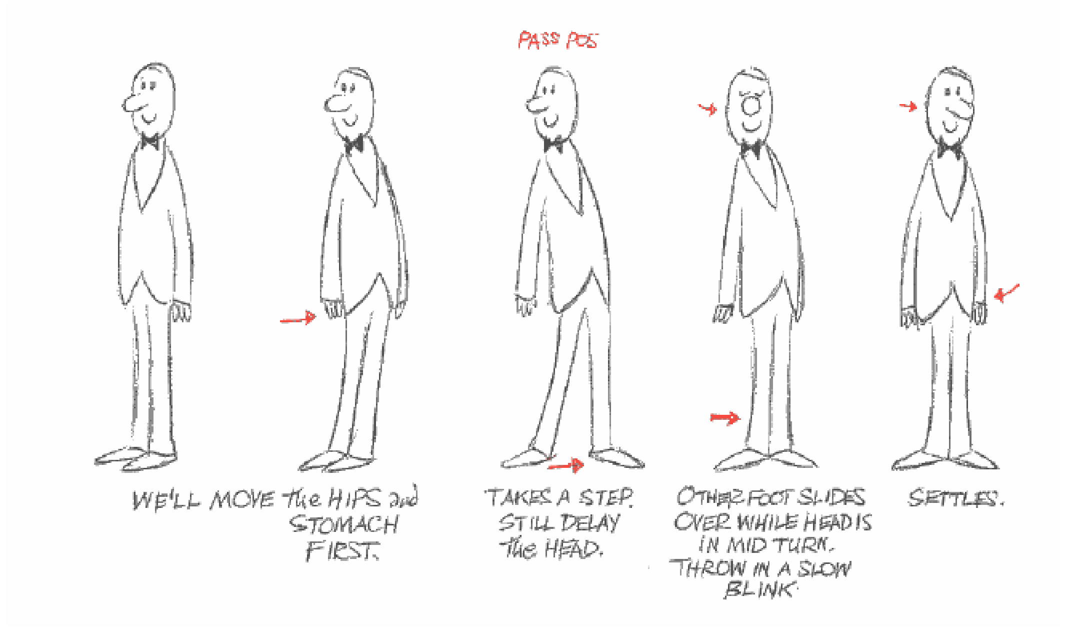
PASS POS
WE'LL MOVE the HIPS and STOMACH FIRST.
TAKES A STEP. STILL DELAY the HEAD.
OTHER FOOT SLIDES OVER WHILE HEAD IS IN MID TURN. THROW IN A SLOW BLINK.
SETTLES.

Figure 2.2: Simple character animation demonstrating follow-through and overlapping action on the arms and legs. Source: The Animator's Survival Kit [36]

2.2.1 Commercial Examples

Critically analyzing character animation, it is easy to find these concepts used often in animation past and present. One exciting use of drag and follow through hearkens back to a style of animation prominent in the late 1920s through 1930s: rubber hose animation. This style, used extensively by Fleischer Studios, re-envisions characters' limbs as rubber hoses without any skeletal joints or muscles. This style allows the character's arms and legs to flail around in the air. Examples can be found in timeless cartoons such as Popeye, Betty Boop, and Felix the Cat.

Many examples can also be found digging into Ub Iwerks's work. A classic example is from Disney's Silly Symphonies' pilot episode, *Skeleton Dance* (Fig. 2.3) [3]. Released in 1929, Ub Iwerks nearly single-handedly finished the animation in roughly six weeks [27]. There are parts of the animation when the bones follow the rubber hose animation style. Individual bones bend to add fluidity to the overall movement. This iconic short cemented Walt Disney's tradition of creating "musical novelties" through the Silly Symphony series. Another similar example, also from Ub Iwerks and Grim Natwick, is from the cartoon series Flip the Frog from the early 1931 episode *Spooks* (Fig. 2.4) [1].

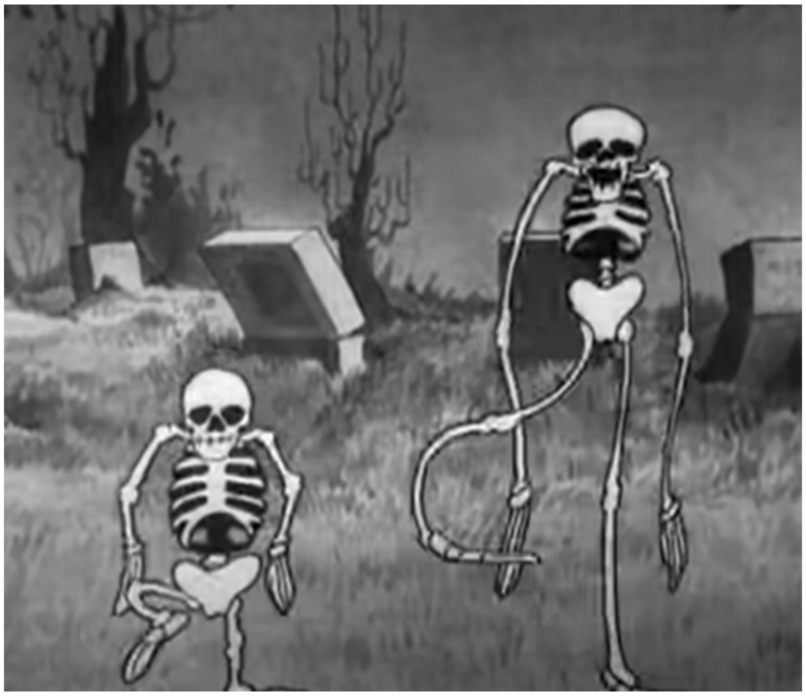

Figure 2.3: Example of a skeleton with rubber-hose bones. Frame from *The Skeleton Dance*, Silly Symphonie by Walt Disney, animated by Ub Iwerks, Les Clark, and Wilfred Jackson, 1929. Distributed by Walt Disney Animation Studios [3].

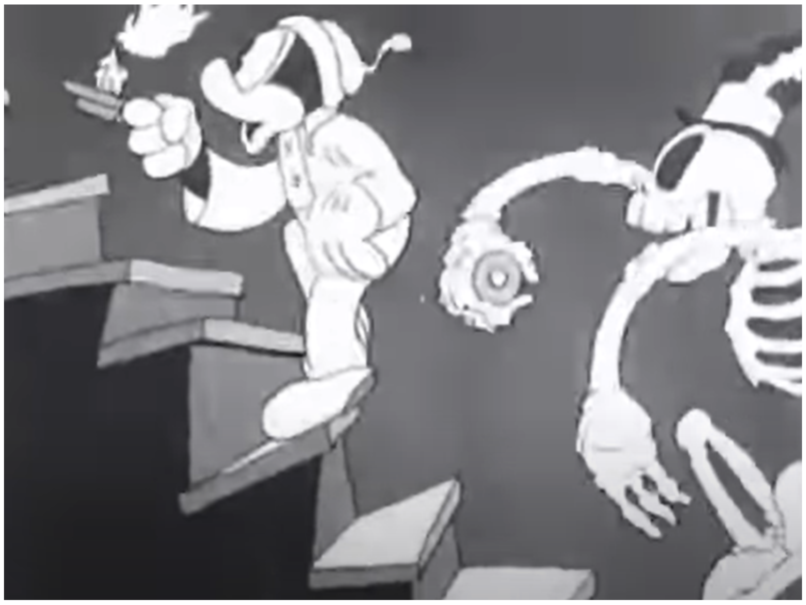

Figure 2.4: Example of a skeleton with rubber-hose bones. Frame from *Spooks*, Flip the Frog by Ub Iwerks and Grim Natwick, 1931. Distributed by MGM [1].

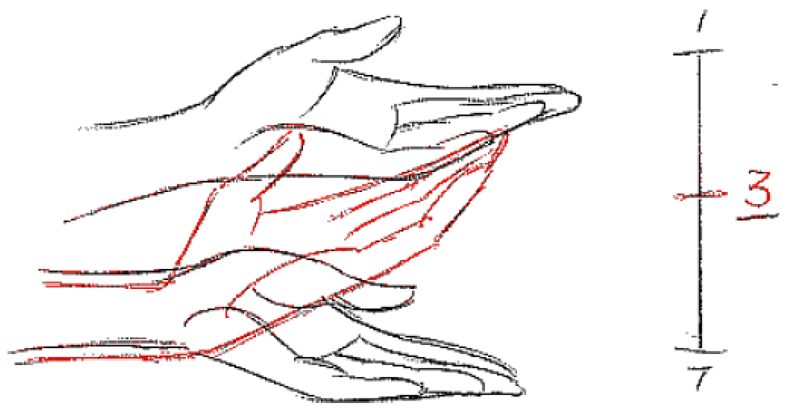

Figure 2.5: Breakdown pose used to add fluidity in the animation without resorting to bending joints or using rubber hose animation. Source: The Animator's Survival Kit [36].

Jumping ahead in time, animators such as Art Babbit and Richard Williams elevated the art form by avoiding such extreme uses of this technique [36]. They instead found ways to apply this method sparingly while still making an impact (Fig. 2.5). R. Williams explains and expands his recommended approach to obtain fluidity in his seminal work "The Animator's Survival Kit" [36]. He suggests retaining the notion of a skeletal structure, but adding flexibility by successively delaying or advancing different parts of the hierarchy.

2.3 Modern Animation Pipeline

2.3.1 3D Mesh

The modern 3D animation pipeline involves numerous steps. To animate a character, it must first be designed and modeled in 3D (Fig. 2.6). The 3D model must consist of a polygonal mesh of vertices. Although other forms of 3D modeling exist, such as NURBS surfaces, these are typically not used in the entertainment industry; even in instances where they are, they are eventually converted to a polygonal mesh. There are certain facilities which exclusively use NURBS, however their use is not as widespread as it once was, especially in the entertainment industry [30]. Once the character is modeled, it exists as a static geometry in 3D space. One can only perform basic transformations on the character, such as translation, rotation, and scale. More complex transformations and deformations require the use of additional techniques and setup.

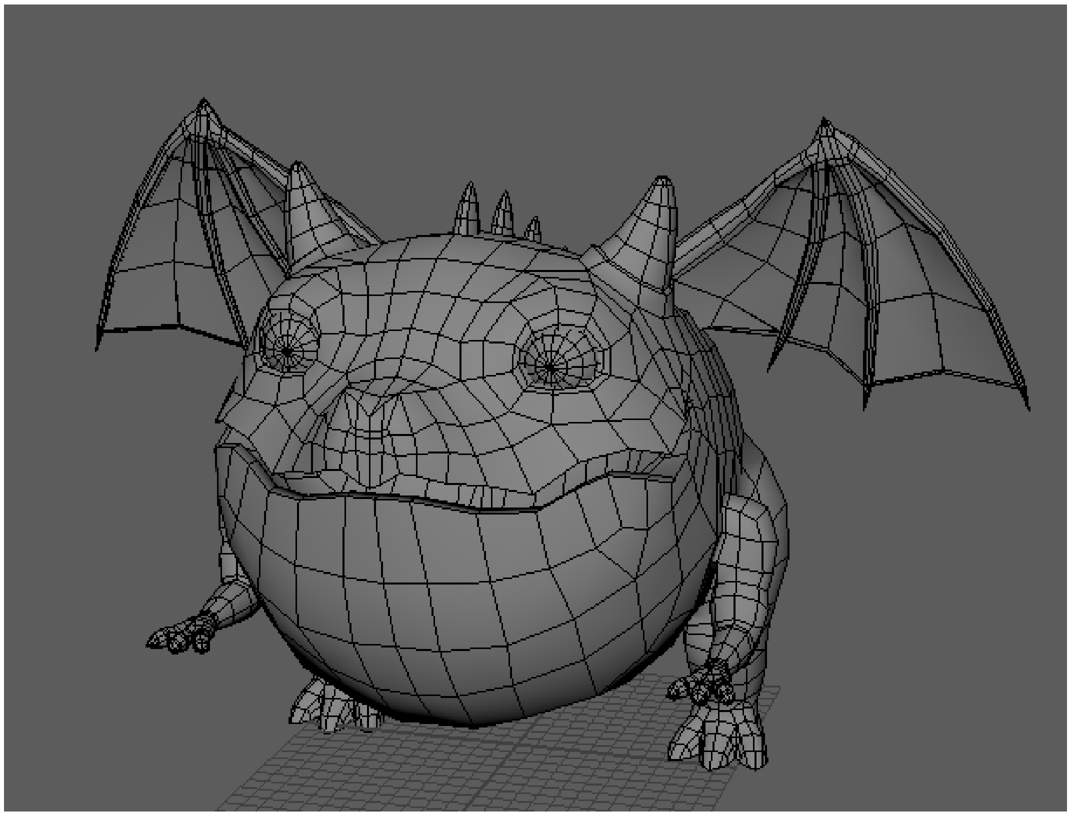

Figure 2.6: Polygonal 3D model of a dragon

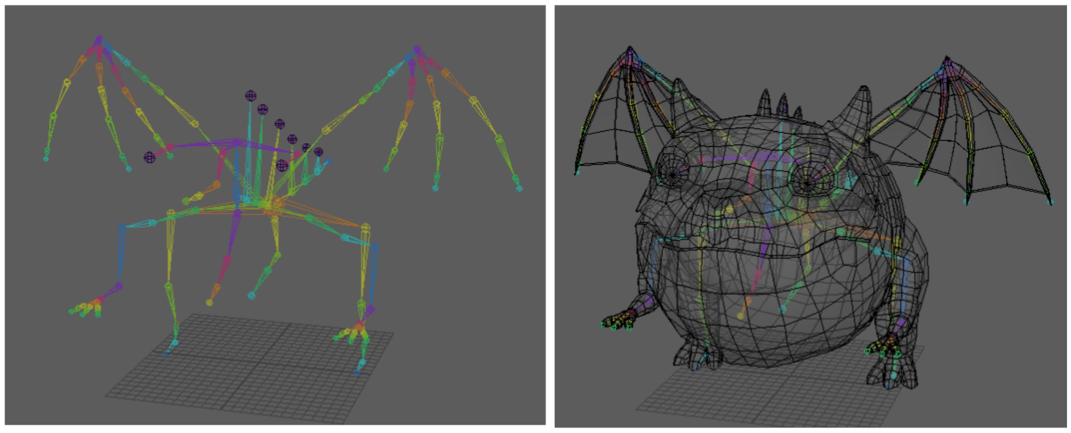

Figure 2.7: Skeletal rig of a dragon model. Rigged by Rodney Costa

2.3.2 Skeletal Rig

For character animation, one must first set up an animation skeletal rig for the character (Fig. 2.7). This skeletal setup consists of a hierarchy of bones/joints. Typically, the position of these joints in 3D space may derive inspiration from the natural anatomy of the character. For example, if it is a human character, the skeletal setup may contain one spine, two arms, two legs, etc. Constraints are then added to limit the rotation of these joints; the elbow or knee joint moves differently than a shoulder joint, for example. One may constrain certain joints in the skeleton to imitate these limits, if so desired.

2.3.3 Skinning

Finally, after the skeleton has been setup as desired, it must be bound back to the character mesh through a process called "skinning". This is done to enable the skeletal movement to deform the character mesh. When an animator moves the forearm of the skeleton, the skeleton should deform the corresponding forearm of the character to match. The simplest form of skinning, rigid skinning, involves rigidly attaching vertices to a single joint, i.e., every vertex can only be attached to one joint each. When a joint rotates, all the vertices attached to it are transformed along with it. This is a very simple approach, but can lead to self-collision, especially in areas like the elbows and knees (Fig. 2.8).

Linear Blend Skinning (LBS) addresses this issue by allowing a single vertex to be influenced by multiple joints. When skinning a skeletal rig to a mesh in this method, artists must take care to

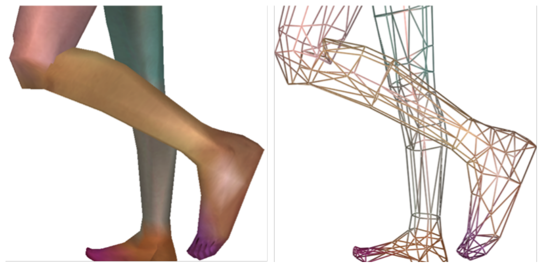

Figure 2.8: Rigid skinning can cause self-collision in areas with high bend angles.

finely tune "skinning weights" on the character. These skinning weights are a sort of mapping to indicate which bones in the skeleton have influence over which vertices on the mesh. For instance, the parts of the mesh around a character's elbow may be influenced by the movement of both the forearm and upper arm. Such intricacies are handled by the mapping through per-vertex weights (Fig. 2.9); each vertex on the mesh is associated with a set of bones which influence it, and each of these influences is given a "weight". This weight determines to what extent the corresponding bone affects the specified vertex, giving us a blend between the deformations dictated by rigid skinning. These weights represent the relative amount of deformation exerted by each bone that influences a given vertex. The total sum of weights on each vertex adds up to one to avoid scaling artifacts.

It is common practice to set up various controllers (typically using curves) around the character to control joint movement (Fig. 2.10). This allows animators to quickly select the curves to control the parts of the body they wish to animate. Once the skinning is complete, one may animate the character by transforming the joint controllers, and adding keyframes to time out the animation as desired. Many of the steps involved in rigging and skinning may be automated using procedural or expression-based techniques [7].

2.4 Technical Advancements in Graphics

While LBS remains a commonly used skinning technique, it is not without flaws. The linear interpolation of transformation matrices used in LBS causes many well known artifacts, including the "candy-wrapper effect" (Fig. 2.11). Recent technical advancements in graphics have improved the process of rigging and skinning characters, providing enhanced quality of deformation. For example, Dual Quaternion Skinning (DQS) is another skinning technique based on the concept of a

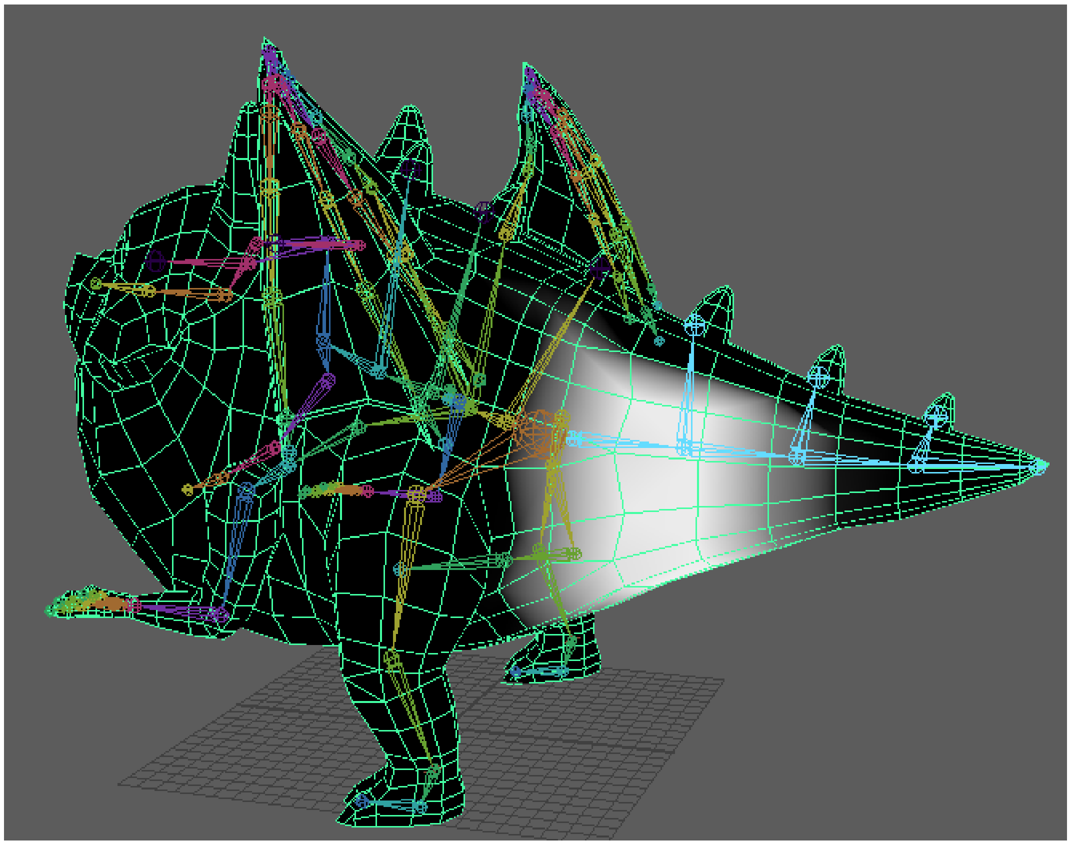

Figure 2.9: Skinning weights of a single joint on the tail of a dragon model. Rigged by Rodney Costa

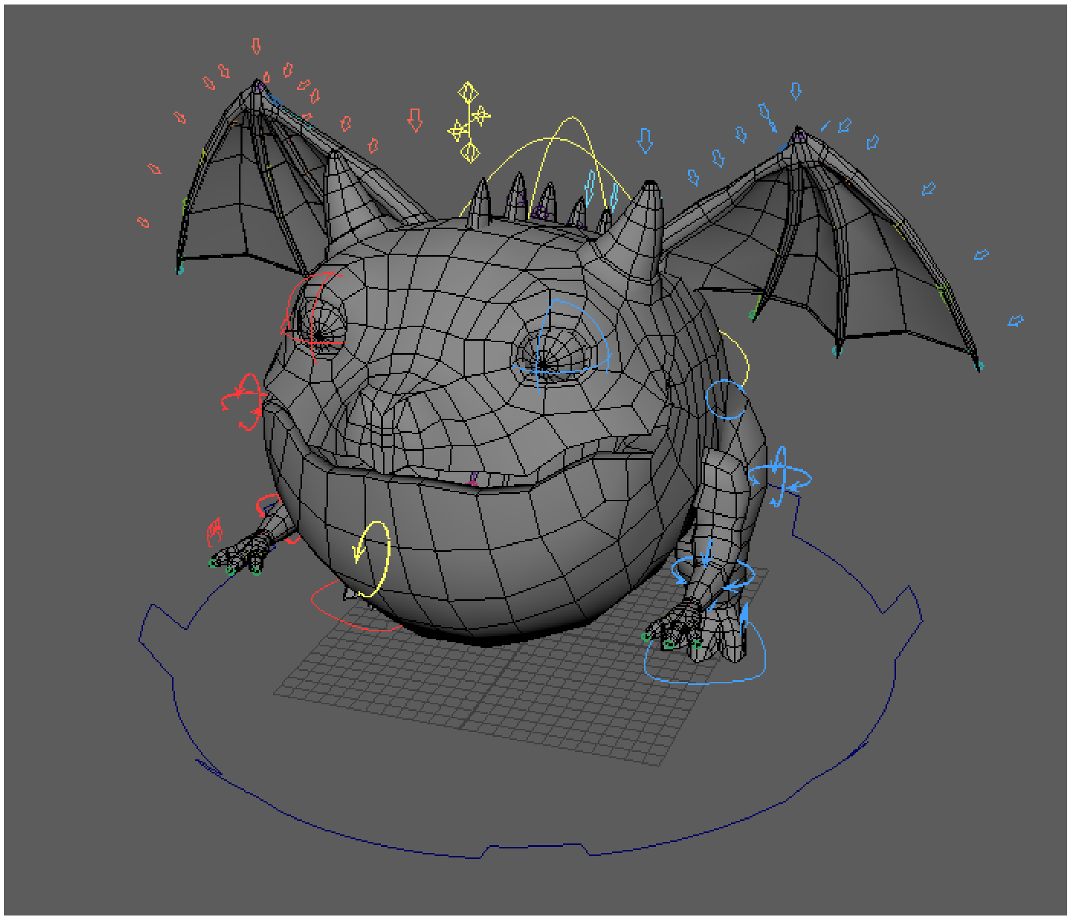

Figure 2.10: Skeletal rig of a dragon model with controllers set up to control the rig. Rigged by Rodney Costa

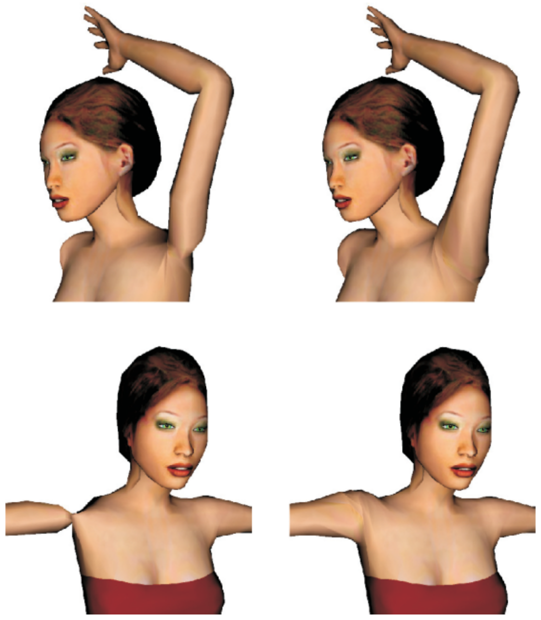

Figure 2.11: Results of LBS (left) and DQS (right) deformation applied on the same skeletal transformation. The pinching of the arms due to extreme joint rotation (*candy wrapper effect*) can be see here. [17]

dual quaternion (a quaternion with "dual number" coefficients) [17]. DQS solves the candy-wrapper issue, as dual quaternion mathematics is better suited to represent such transformations. Further extensions to DQS have also been proposed to make it more suitable to the production requirement of the animation industry [23].

Many of these methods require meticulous efforts from the rigger to manually setup paint weights and check for unappealing deformation through a trial and error process. In scenarios where such deformation is not directly avoidable, riggers may resort to the use of corrective blend shapes to create more appealing deformation. While this is an effective method of addressing the issue, it further consumes the artists' time by requiring the respective poses to be manually resculpted for use as blend shapes. Further methods were developed to ease this burden. Delta Mush is one such example developed by Rhythm and Hues [25]. Delta Mush (DM) operates on top of a rigidly skinned mesh. It iteratively smooths the geometric mesh (using Laplacian smoothing) on each frame to reduce self-penetrations. This leads to a loss of surface detail. The detail is then restored by transforming the mesh using precomputed delta values which reflect changes between the smooth and unsmooth meshes. While DM is an effective technique, its iterative nature hinders its use in real-time applications such as virtual reality or gaming. More optimized versions of DM, such as Direct Delta Mush (DDM) offer solutions to this problem by removing iterative computations and operating on the skinning weights, rather than on the mesh itself [22].

Other approaches to easing the process of creating a rig include Implicit Skinning [34]. This technique approximates the rigged mesh as an iso-surface. It generates a set of implicit fields around each skeletal joint, and uses the motion of these fields to guide vertex deformations. This method approximates skin-contact effects and creates aesthetically pleasing muscular bulges with real-time deformation. Later enhancements expand the gamut of deformations possible through this method by accounting for skin elasticity (effect caused by the skin sliding during joint rotation) [35].

Other methods, including "Fast Automatic Skinning Transformation", attempt to enhance skin deformation by using energy based systems. Rather than having an artist or rigger extensively specify skinning transformations and constraints, this technique allows animators to work with less degrees of freedom and automatically computes the remaining degrees of freedom using an approximated energy-based formulation [18]. Similar to energy-based systems, physically-based deformation techniques are available in plenty. These methods, unlike most geometric approaches, are capable of automatically augmenting primary deformation of skeletal animation by simulating secondary motion and volume conservation. From generating a volumetric mesh to approximate tissue, muscle, and fat motion [13] to using force dynamics as the building blocks of rigging systems [9], these methods exhaustively cover novel approaches to many of the problems present in geometric deformation systems, such as collision detection. More computationally-friendly simulation techniques enable real-time deformations for interactive gaming. One such method converts the character mesh into multiple layers of voxels, each layer corresponding to an anatomical phenomena such as bone, muscle, fat, etc. Applying primary deformation on the bone voxels and propogating this motion through neighboring voxel layers automatically generates secondary motion [28].

Certain simulation systems also attempt to use physically-based techniques to create more cartoon-like deformation. Such models approach this problem in their own unique ways. One utilizes the velocity of the rigid body to dynamically simulate squash and stretch deformation [15]. Another method focuses efforts not only on the effective deformation of animated characters, but also on enabling artists to *control* these deformations with ease [12]. *Complementary Dynamics*, a very recent work in this area of study, proposes a method to use dynamics to add secondary motion to a rig. Unlike other existing approaches, this method preserves the primary action animated by the artist, and uses physics to simulate complementary secondary action. This secondary action is simulated in a subspace orthogonal to the primary action to enable the resulting deformation to enhance the artists' intended animation, rather than undo it [38]. Another work proposes generating implicit surfaces and applying forces to create cartoon-like deformation, including squash and stretch, and followthrough [31]. One method in motion capture emulates cartoon-like rubber-hose animation from traditional animation by allowing joints to grow in length. The skeletal hierarchy of the rigged character is divided into "sub-joints" with dynamic springs interconnecting them. The sub-joint positions are then computed using Bezier curve interpolation to create smoother deformation, resulting in the character having curved limbs [19]. "Kinodynamic Skinning" describe skin deformation by using skeletal motion to induce velocity-based vector fields [5]. This use of dynamics enables volume preservation and prevents unwanted "fold-over" trajectories.

The work presented in this thesis, however, uses a geometric approach, not a dynamic one. We work with instantaneous values of velocity and acceleration, i.e, kinematic motion, to describe and drive the deformation of the skinned mesh, rather then integrating over a time step. Methods closer to this approach include the work by Noble et. al. [29], which attempts to curves character limbs by applying non-linear deformers onto joints. The targeted joints and bend angles are informed by the character animations "lines of action". Likewise, the squash-and-stretch deformer by Kwon et al. [20] proposes a geometric approach to generate squash and stretch motion by deforming the joints themselves. This controlled scaling of skeletal joints cuses the skinned mesh to deform accordingly and creates a result that is closer to the animation style found in traditional animation [20].

Unlike the above stated works, the methods described in this thesis utilize the kinematic motion of the skeleton to inform the deformation of the mesh, rather than deforming the skeletal joints themselves. Noble et. al. [29] determines the lines of action on their characters using a method based off of bone velocity, which is similar to our approach as well. However, their approach considers a targeted joint and its parent in isolation. The overall hierarchy of the skeleton is not considered in this method. Also, the resulting deformation is finally applied onto the target mesh through the use of Autodesk Maya's lattice deformers and non-linear bend deformers. In contrast, our approach is to compute effect-specific deformation functions that directly define the vertex displacement resulting from the deformation. No additional deformers are required to visualize the final effect.

Chapter 3

Velocity Skinning

Velocity Skinning is a recent technique in computer graphics. It is a collaborative effort of multiple researchers, including myself. While this work is not the focus of this thesis, the two are tightly coupled with each other. This chapter is here to provide more clarity on the topic. My personal contributions to this publication are detailed in sections 3.4 onward. Further details about Velocity Skinning may be found in the publication [32].

3.1 Motivation

When a character is animated, the underlying skeletal joints translate and rotate, causing the character mesh to deform, creating the illusion of animation. When these joints move in space, their positions and orientations change over time. If we compute the rate of this displacement, we can obtain the corresponding velocity of the joint. Velocity Skinning presents a method to exploit this velocity information inherently present in skeletal animation to aesthetically deform the character mesh to which the skeleton is skinned. Put simply, the velocity of a joint (and its ancestry of joints) is used to displace the vertices influenced by it on the character mesh. This displacement can be parameterized to create the desired effects. Further details about the method are provided in subsequent sections. The objectives of this method are:

- To automate the process of creating squash and drag effects
- To work out of the box with user controllable parameters

- To work in real-time on meshes with up to 10^6 triangles
- To perform non-linear time edits, i.e., to be able to create the desired effects without requiring any information about the history of the deformation

3.2 Method

3.2.1 Formulation

We begin with the standard Linear Blend Skinning (LBS) method formulation. In LBS, for every resultant vertex position p on the mesh, we have

$$p = \sum_j b_j p_j.$$

Where j is the collection of joints influencing the vertex position p,

b_j is the skinning weight,

p_j is the vertex position obtained from rigid skinning

Differentiating with respect to time, we get

$$v(p) = \dot{p} = \sum_j b_j \dot{p}_j.$$

Here $\dot{p}_j$ is the net velocity of the vertex position p_j. This depends on the rigid motion of joint j. However the the rigid motion of joint j is affected by the rigid motion of its parents, and so on and so forth. Therefore, we can further decompose $\dot{p}$ along the hierarchy of joints.

$$\dot{p}_j = \sum_{k \in Anc(j)} v_k$$

Where v_k is the relative velocity induced by joint k with respect to its parent. Plugging this decomposition into the previous equation, we have

$$v(p) = \sum_j b_j \left(\sum_{k \in Anc(j)} v_k \right).$$

Reformulating this as a single summation, we get

$$v(p) = \sum_j \tilde{b}_j v_j$$

where

$$\tilde{b}_j = \sum_{k \in Desc(j)} b_k.$$

Based on the above equations, we can see that the effective velocity of the position p is a linear sum of the local velocities of its relative joints, i.e., its ancestry of joints. The final formulation shows that the velocity skinning weights $\tilde{b}_j$ are directly derived from the rigid skinning weights b_k. This notion of a local joint velocity may be further decomposed into translational and rotational components.

$$v_j(p) = u_j + \omega_j \times (p - p_j)$$

where

u_j is the velocity of joint j due to translation,

ω_j is the angular velocity of joint j,

$p - p_j$ is the vector connecting p to the center of rotation,

$\omega_j \times (p - p_j)$ is the effective angular velocity at p due to rotational motion (Fig. 3.1)

Putting everything together, we have

$$v(p) = \sum_j \tilde{b}_j \left(u_j + \omega_j \times (p - p_j)\right). \tag{3.1}$$

3.2.2 Deformation

Equation (3.1) gives us the velocity at position p. However, what we require is the deformation at position p caused by this velocity $v(p)$. Just as $v(p)$ is related to the linear sum of relative velocities of its ancestor joints, let us suppose that the resulting global deformation is also related to the linear sum of relative local deformations of its joint hierarchy. The final deformation applied onto point p is

$$p_{final} = \underbrace{p}_{skinning} + \underbrace{d(p)}_{velocity-skinning},$$

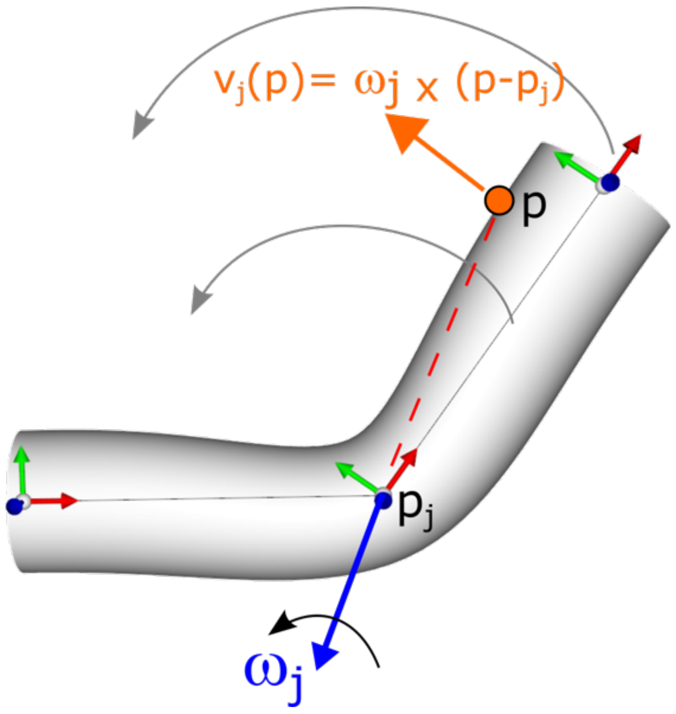

Figure 3.1: Visualizing rotational component of Velocity Skinning formulation

where $d(p)$ is the global deformation at point p;

$$d(p) = \sum_j \tilde{b}_j \psi_j(p), \tag{3.2}$$

and $\psi_j(p)$ is the local deformation of p caused by the joint j.

Like the velocity term, this local deformation can be further decomposed into deformation caused by translational and rotational components:

$$d(p) = \sum_j \tilde{b}_j(\psi_{u_j} + \psi_{\omega_j}(p)).$$

This formulation gives us a linear blend of deformations which propagate along the hierarchy of the skeleton. However, ψ is merely a deformation function. It may refer to any deformation. In this context, our objective is to model drag and squash. Therefore, we require a deformation function for drag, and a separate one for squash. Conceptually, for the drag deformation, we simply translate or rotate the position p opposite to the direction of motion. Skipping ahead, we have the following formulation for the drag deformer ψ^{dv}:

$$\psi_j^{dv}(p) = (\psi_{u_j}^{dv} + \psi_{\omega_j}^{dv}(p))$$

$$\psi_{u_j}^{dv} = -u_j$$

$$\psi_{\omega_j}^{dv}(p) = (R(\theta) - I)(p - p_j) \tag{3.3}$$

where

$R(\theta)$ is a rotation matrix representing the rotation of joint j by angle θ,

I is the identity matrix, and

p_j is the vertex position obtained from rigid skinning.

In the above equation, R is constructed using the angle of rotation *theta* about the axis defined by ω

$$\theta = -k_{dv}||v_i|| \tag{3.4}$$

where

v_i is the vertex velocity, and

k_{dv} is a user controlled parameter defining the magnitude of deformation.

Similarly, we have the squash deformers

$$\psi_j^{sv}(p) = (\psi_{u_j}^{sv} + \psi_{\omega_j}^{sv}(p)) \tag{3.5}$$

The squash deformation involves generating scaling matrices. This is done by computing a *centroid* for the application of the scaling deformation, which is computed as the barycenter of the vertices influenced by a given joint. Like the drag deformer, the squash deformer includes a user controllable value k_{sv} to scale the intensity of squash deformation. Further explanations regarding these deformation functions and their derivations can be found in the relevant publication [32].

3.3 Per-Vertex Paint Weights

Till now, we have discussed different effects that velocity skinning can produce. We have seen that artists may tweak the input parameters k_{dv}, k_{sv}, etc. to define how subtle or pronounced

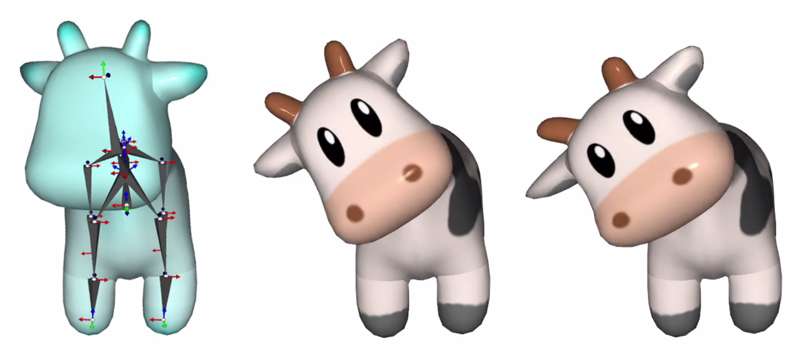
Figure 3.2: Effects of weight painting: Vertex weights on a cow model painted to emphasize deformation on the horns and ears (left), LBS results (center), Velocity Skinning results with weight painting applied (right)

the deformation should be. However, artists may wish to have more granular control over the effects. It is common to wish for effects to be more pronounced in certain parts of the character than others. This can be achieved through the use of per-vertex weight painting. These weights are simply scalar values assigned to each vertex on the mesh, used to scale the deformation effect applied on said vertex. Artists may modify these weights non-uniformly over the mesh to create a more fine-tuned result. This allows artists to localize the effects to specific parts of the mesh. For example, in Fig 3.2, the weights on the mesh were painted to emphasize the deformation to the cow's head, and more specifically on the ears and horn. Such granular control cannot be expressed through pure skeletal deformation. From the user interface, these per-vertex weights may be interactively painted over the surface of the mesh.

The artists are not only limited to positive scalar values. If so desired, they may use negative weights to cause the vertex to deform in the opposite direction. This may be useful if you require parts of the skin to *lean in* to the animation rather than drag behind it. For example, consider the bird shown in Fig 3.3. This rig has just a single joint. All of the deformation seen is created through the use of paint weights. We see that the while most of the body drags behind the animation, one of the wings *leads* the action.

Bear in mind that while using per-vertex paint weights, the aesthetic quality of the resulting deformation lies in the hands of the user. Great care must be taken to paint the weights. The onus is on the user, much like in LBS and DQS. For example, consider the deformation of the dragon shown in Fig 3.4. Discontinuity is visible in certain parts of the mesh. This is a direct result of the

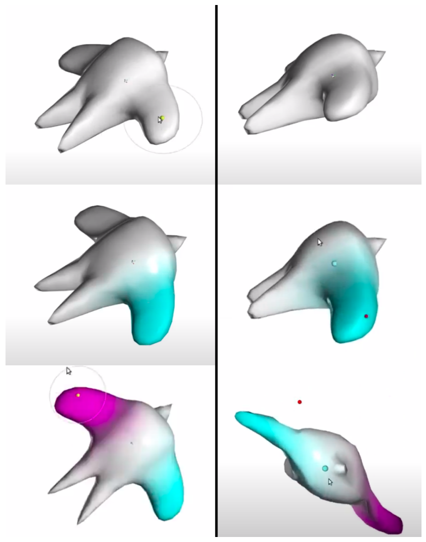

Figure 3.3: Weight painting: No weights applied (top row), positive weights applied (center row), positive and negative weights applied (bottom row) [32]

weight painting. The change in vertex weights across the mesh is too sudden to achieve a smooth deformation result. However, if the artist wished to create, say, a zombie dragon with parts of the skin peeling off and flapping around, this deformation may be deemed usable. At the end of the day, as with all art, context determines what is good and bad. It is the prerogative of the artist to exploit these tools as they see fit.

3.4 Enhancing Artist Control

3.4.1 Twist-Bend Decomposition

In a practical scenario, artists may desire enhanced control over the previously discussed deformation effects. For example, let us assume that we have an aged character with weakened and saggy muscles. When this character holds up their arm, we can see the clearly defined muscular degeneration. If they rapidly twist the arm from side to side, the sagging muscles will flap back and forth. However, if they rapidly bend their arm at the same pace, the magnitude of muscle movement is not nearly as high. Therefore, we may assume that artists may want to handle different types of rotational movement differently. One approach to such enhanced control is to decompose rotation due to bending and rotation due to twisting.

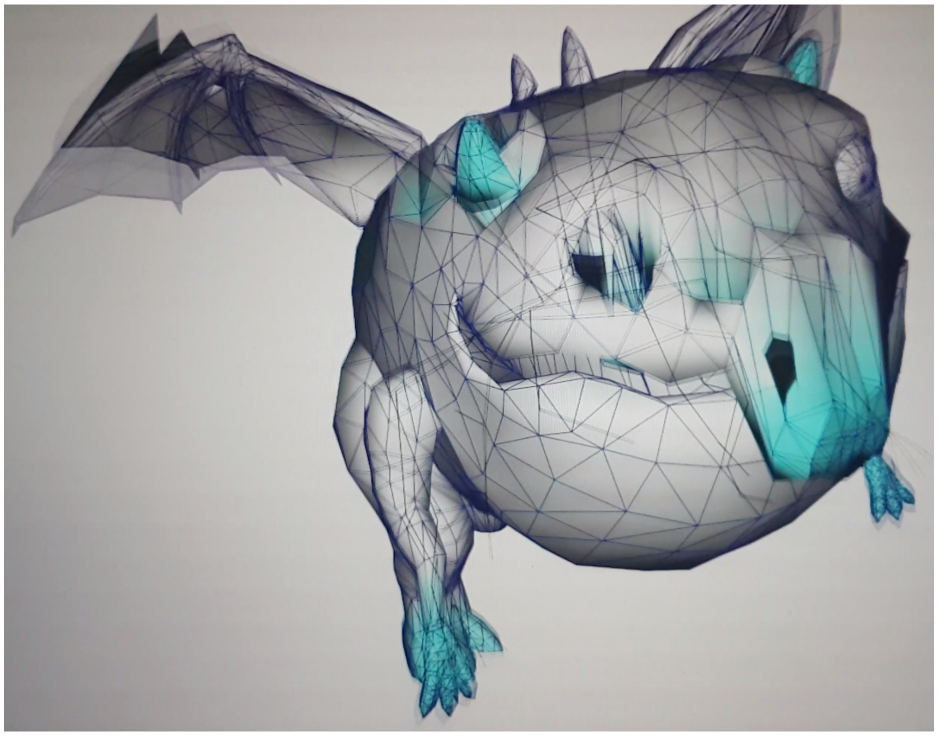

Figure 3.4: Effects of mesh deformation from discontinuous paint-weights

Mathematically, the axis of rotation of a bone is give by ω_j. In the case of bending motion, we know that the bone must lie on the plane orthogonal to ω_j. However, considering the case of twisting motion, the bone is aligned with the axis of rotation ω_j (see Fig 3.5). Therefore, we can decompose these two different motions by comparing the axis of rotation to the joint orientation. When they are aligned, we have a pure twisting motion. When they are orthogonal, we have pure bend. Any combination of these two motions (bend and twist) can be represented as a weighted sum of the two motions. For convenience, we consider a weighted linear sum.

$$twist + bend = 1$$

This alignment can be computed using a dot product as:

$$m = |\hat{\omega_j} \cdot l_j| \tag{3.6}$$

$$m \in [0, 1]$$

where

m is the *twist magnitude* of the rotation,

l_j is a unit vector aligned with the axis of joint j, and

$\hat{\omega_j}$ is the axis of rotation of joint j.

We can see from this formulation that the twist magnitude m has a value ranging from zero to one, representing the fraction of the net rotation caused by twist. Similarly, we can say that whatever motion is not caused by twist, must be induced by the bending of joints. Therefore, we consider *bend magnitude* $= 1 - m$. These values may be used to scale the net rotation deformation independently, thereby providing a clear decomposition of rotation caused by twist vs bend.

$$\psi_{\omega_j}^{dv'} = \psi^{tv} + \psi^{bv} \tag{3.7}$$

where

$\psi_{\omega_j}^{dv'}$ is the enhanced or updated drag deformer,

ψ^{tv} is the drag due to twist,

ψ^{bv} is the drag due to bend.

We know from (3.4) that θ contains a user controlled parameter k_{dv}. This value may

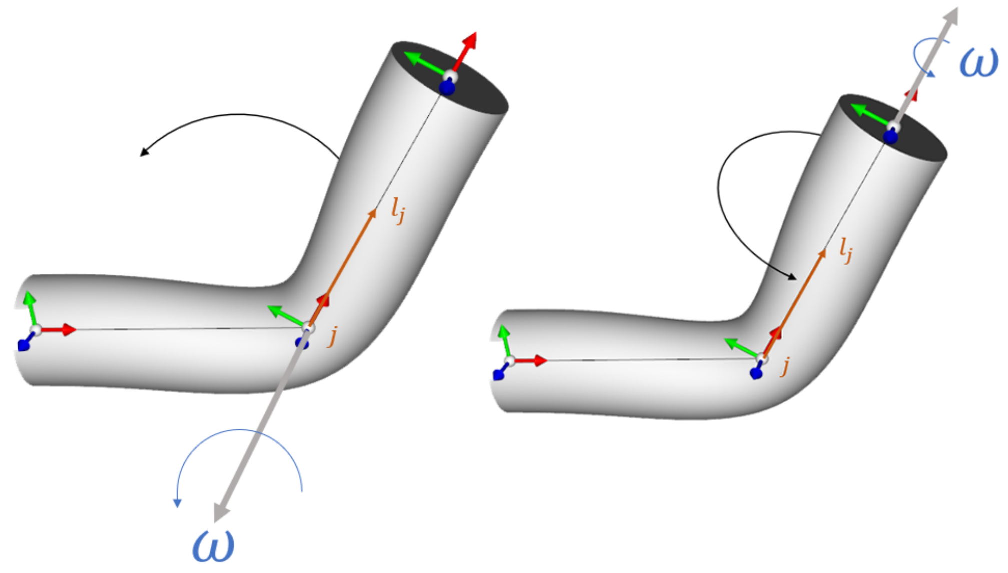

Figure 3.5: Distinguishing rotation due to bending (left) and twisting (right) motions. ω is orthogonal to l_j in the case of bend, and aligned to it in the case of twist.

be scaled independently in the twist/bend deformation functions using the respective magnitudes previously computed. Expanding on (3.3), we can re-envision $\psi_{\omega_j}^{dv}$ to include a parameter for the angle of rotation θ as

$$\psi_{\omega_j}^{dv}(p, \theta) = (R(\theta) - I)(p - p_j). \tag{3.8}$$

Combining (3.6), (3.7), and (3.8) we have

$$\psi_{\omega_j}^{dv'}(p) = \psi_{\omega_j}^{dv}(p, \theta t) + \psi_{\omega_j}^{dv}(p, \theta(1 - t)).$$

We now have a distinct twist-bend decomposition. However, the objective is to enhance artist control over this motion. With the above equation, although we have a decomposition, we do not have any means for an artist to control these twist/bend values independently. Hence, we can reparametrize as

$$\psi_{\omega_j}^{dv'}(p) = \psi_{\omega_j}^{dv}(p, \theta_{tv}) + \psi_{\omega_j}^{dv}(p, \theta_{bv}) \tag{3.9}$$

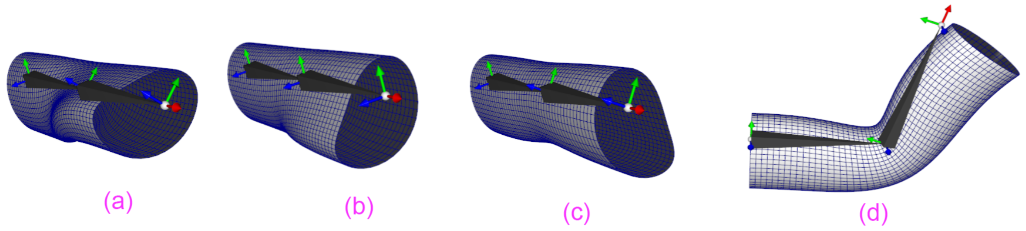

Figure 3.6: Cylinder rigged with the joints off-center demonstrating twist deformation (a, b, and c) and bend deformation (d)

where

$$\theta_{tv} = -mk_{tv}||v_i||,$$

$$\theta_{bv} = (1+m)k_{bv}||v_i||,$$

where k_{tv} and k_{bv} are user controlled parameters defining how pronounced the drag effect caused by twist and bend must be respectively. In Fig. 3.6, one can see the drag deformation effect caused due to the twisting motion vs the bending motion of the cylinder.

3.4.2 Local Joint Control

3.4.2.1 Motivation

Till this point, all of the user controllable parameters discussed, such as k_{dv}, k_{tv}, k_{bv}, etc, have been global controls. It may be beneficial to also consider localizing these parameters to a joint specific control. Artists may find such control desirable to isolate the deformation effects to a specific joint or chain of joints. For example, if we have a character walk cycle, we may wish to localize the deformation to the arms alone. Hence, it would be desirable to control deformation parameters at a per-joint level.

3.4.2.2 Method

Recall the deformation equation 3.2

$$d(p) = \sum_j \tilde{b}_j \psi_j(p).$$

Regardless of what the targeted deformation function is (ψ^{dv}, ψ^{sv}, etc), we know that there exists a value k which allows the user to scale the magnitude of deformation. This is a global value, constant throughout the skeletal hierarchy. Let us replace this with a value k_j local to each joint.

If we consider the case of $\psi^{dv}_{\omega_j}$, we have the angle of rotation θ computed using the global scale k_{dv} as

$$\theta = -k_{dv}||v_i||.$$

Substituting this with the localized value k_{dv_j}, we have

$$\theta = -k_{dv_j}||v_i||.$$

Similar substitutions may be performed for all deformation functions ψ. Note that the high level formulation of velocity skinning remains intact. The net deformation of a vertex position p still depends on the deformation of its ancestor joints. This gives rise to an interesting behavior; it is not possible for a given joint j_i to remove or negate the deformation inherited from its parent joint j_{i-1}. Deformation effects cumulatively propagate down the skeletal hierarchy. This remains consistent with the behavior of the Velocity Skinning method.

This behavior may be better conceptually understood by thinking of it in the context of frames of reference. If a character rotates their shoulder, that will automatically move their elbow, wrist and fingers. Even if they keep their elbow static (without any rotation), the position and orientation of the forearm is still affected by the shoulder rotation. In other words, the forearm is static in a local frame of reference with respect to the elbow joint. However, it is displaced in a global frame of reference. This is the same concept behind the local joint control in Velocity Skinning. Disabling deformation of a joint j_i merely prevents any deformation within the local frame of reference. However, any deformations inherited from higher up the skeletal chain will be preserved, and cannot be negated.

3.4.2.3 Comparison with Paint-Weights

Consider using Velocity Skinning to deform a character walk cycle. We may wish to disable deformation on the legs, but retain deformation on the arms. Although this may be achieved using the per-vertex paint weights discussed in section 3.3, having a joint level control provides yet another

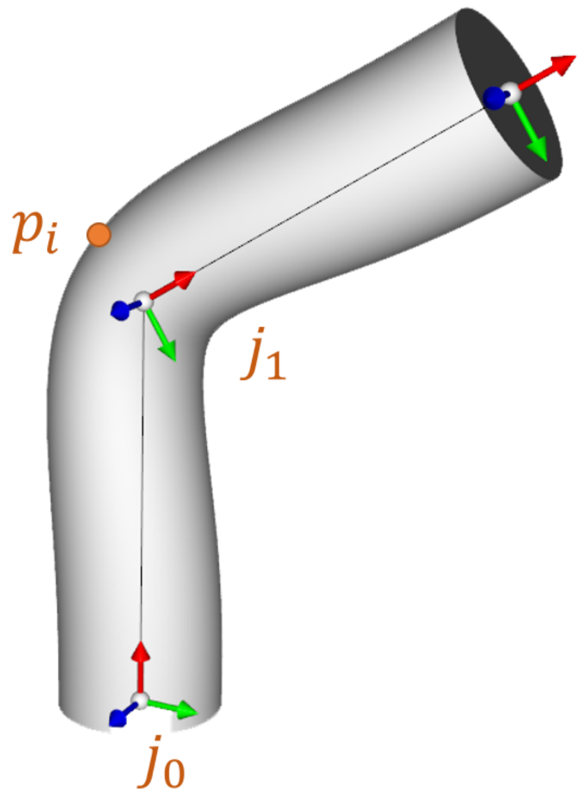

Figure 3.7: Cylinder rig demonstrating a joint chain for the given scenario

approach which may be more desirable in circumstances such as this. If we explicitly know that we do not want a certain joint to have any deformation, it would be much easier to disable deformation on a per-joint level as opposed to manually zero-ing the vertex weights on all relevant vertices. Also, some vertices may be influenced by multiple joints. In such cases, modifying the vertex weights will compute the net deformation from the entire joint hierarchy, and then proceed to scale the result by the artist specified weight. However, as using joint specific control enables artists to negate the deformation of certain joints, it is possible for such vertices to still deform due to the influence of other joints.

Consider the joint chain given in Fig. 3.7. Let as assume that an artist wants to disable deformation on the bottom half of the mesh but have the top half deform. They may use paint weights to disable deformation of the vertices on the bottom half of the mesh and leave the top half untouched, but they must then carefully handle the vertices in the intermediate region, such as the vertex p_i. If the painting is not a smooth gradation, it may lead to undesirable discontinuity in the deformation. However, if approached using local joint control, one may disable deformation of joint j_0 while preserving deformation on joint j_1. In this case, using the joint-level parameterization enables automatically taking advantage of the smoothness and spatial influence of the precomputed skinning weights without requiring any extra set-up or painting.

$$d(p_i) = \sum_j \tilde{b}_j \psi_j(p_i)$$

$$d(p_i) = \tilde{b}_{j_0} \psi_{j_0}(p_i) + \tilde{b}_{j_1} \psi_{j_1}(p_i)$$

$$\psi_{j_0}(p_i) = 0$$

$$\therefore d(p_i) = \tilde{b}_{j_1} \psi_{j_1}(p_i)$$

As we only consider the influence of $\tilde{b}_{j_1}$, the smoothness of the resulting deformation across this region depends on the skinning weights b_j. If appropriate care has been taken when painting these skinning weights, then Velocity Skinning naturally creates a smooth transition on the mesh across the vertices between the two joints. This is not extra work for the artist, as the standard industry pipeline demands fine-tuned paint weights for most commonly used skinning approaches. Thus, for this particular scenario, using local joint control may provide a more pleasing deformation out of the box with less artist intervention. Of course, the artist may use a combination of both of these approaches if they so desire.

Chapter 4

Acceleration Skinning

4.1 Analysis of Acceleration

We know from Chapter 1 that followthrough is a fundamental principle of animation and has been used extensively in animation throughout history. Modeling this effect using a framework like Velocity Skinning could be beneficial to artists; drag and followthrough are two effects that often go hand in hand with each other. Modeling this effect using the existing velocity skinning framework could create a more wholesome and unified tool.

Breaking down the desired followthrough effect into a set of technical requirements reveals that to successfully create this effect, the character mesh must be deformed beyond the skeleton's rest position, ie, the mesh must deform even after the skeletal animation has come to a stop. This requires re-envisioning our existing concepts. When a rigid body is at rest, its speed and velocity are zero. If there is no motion, there is no velocity. Acceleration, however, gives us more ways to break down the motion. When a rigid body is perfectly at rest, its acceleration is zero. However, if it is in motion and then comes to rest, there exists a deceleration value acting upon it even after the body has come to rest. This deceleration is responsible for bringing the body to a stop. It exists for a period of time, and eventually tapers to zero as well.

Fig. 4.1 demonstrates this relationship between velocity and acceleration. This graph plots the Z-axis component of angular velocity ω and angular acceleration α for a rotating object. This object begins rotating from rest, rotates for a while, and comes to rest again. We see from the graph that the ω term has only one peak (or lobe). In contrast, the α value has two lobes, one positive and

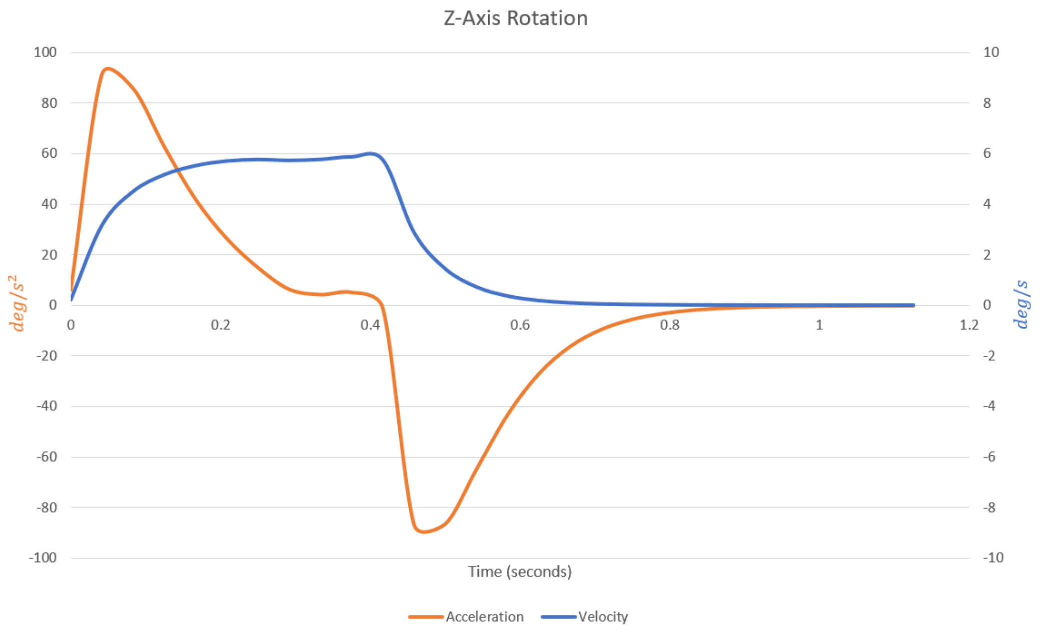

Figure 4.1: Relationship between angular velocity ω and angular acceleration α for a body that begins rotating and returns to rest visualized on a graph. This graph plots only the Z-Axis component of these two terms.

one negative. Similar behavior may be observed for linear motion. We use this to our advantage to model followthrough; The positive lobe is used to deform the mesh in one direction and the negative lobe is used to deform the mesh in the opposite direction.

Mathematically, the velocity of a rigid body can be described using the sum of its two quantities: linear velocity u, and angular velocity ω. If we possess knowledge of the body's center of mass or center of rotation, then these two quantities are sufficient to compute the velocity at any point on the surface of the body. The velocity at a point p on the surface of the body is defined as

$$v = \dot{p} = u + \omega \times r$$

where r is the relative position of p with respect to the axis of rotation. Similarly, we can extend this logic to acceleration as well. The acceleration of a rigid body may be described using a sum of its linear acceleration a and angular acceleration α, where

$$a = \dot{u}$$

$$\alpha = \dot{\omega}.$$

The local acceleration experienced by a point p on the surface of the body can, similarly, be expressed as

$$LocalAcceleration = \ddot{p} = a + \alpha \times r + \omega \times \omega \times r.$$

Relating this concept back to the topic of a skeletal rig, this *LocalAcceleration* can be used to represent the acceleration of a vertex on the mesh. The local center of rotation is merely the axis about which a given joint rotates. Notice that when differentiating vertex angular velocity $\omega \times r$, we end up with two quantities $\alpha \times r$ and $\omega \times \omega \times r$.

- $\alpha \times r$: We call this term "tangential acceleration" a_t(Fig. 4.2). It acts in a direction tangential to the circle of rotation at the vertex point of p, and is parallel to the the vertex velocity vector $\omega \times r$.
- $\omega \times \omega \times r$: We propose to call this term "centripetal acceleration" a_c(Fig. 4.2). It acts along the radial direction, pointing inwards, towards the center of rotation, its magnitude is proportional to the distance from the center of rotation.

Fig. 4.3 visualizes these terms on a rotating flower. For convenience, we rephrase the above terms as functions:

$$a_t(\alpha, r) = \alpha \times r \tag{4.1}$$

$$a_c(\omega, r) = \omega \times (\omega \times r) \tag{4.2}$$

4.2 Incorporating Acceleration in Skinning

Consider an animated clip where a character begins to wave their arm and then stops mid-wave. Suppose we would like to apply followthrough onto this animation. When the arm stops animating, rather than coming to an abrupt stop, we would like it to shoot beyond the stopping pose and then settle back into it, as discussed in section 2.2. Suppose we would like to model this as an effect using Velocity Skinning.

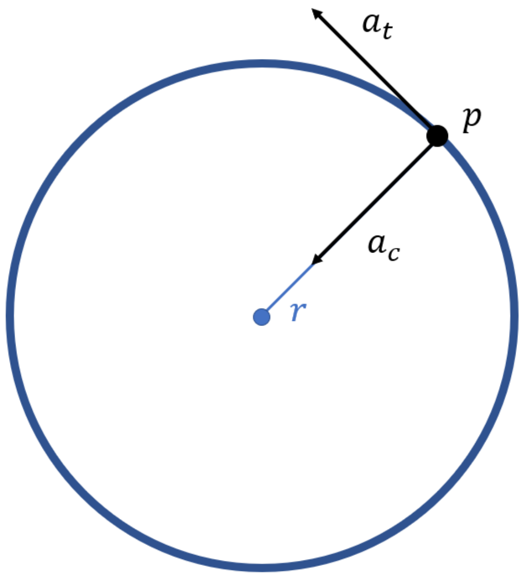

Figure 4.2: Centripetal and tangential acceleration terms visualized for circular motion

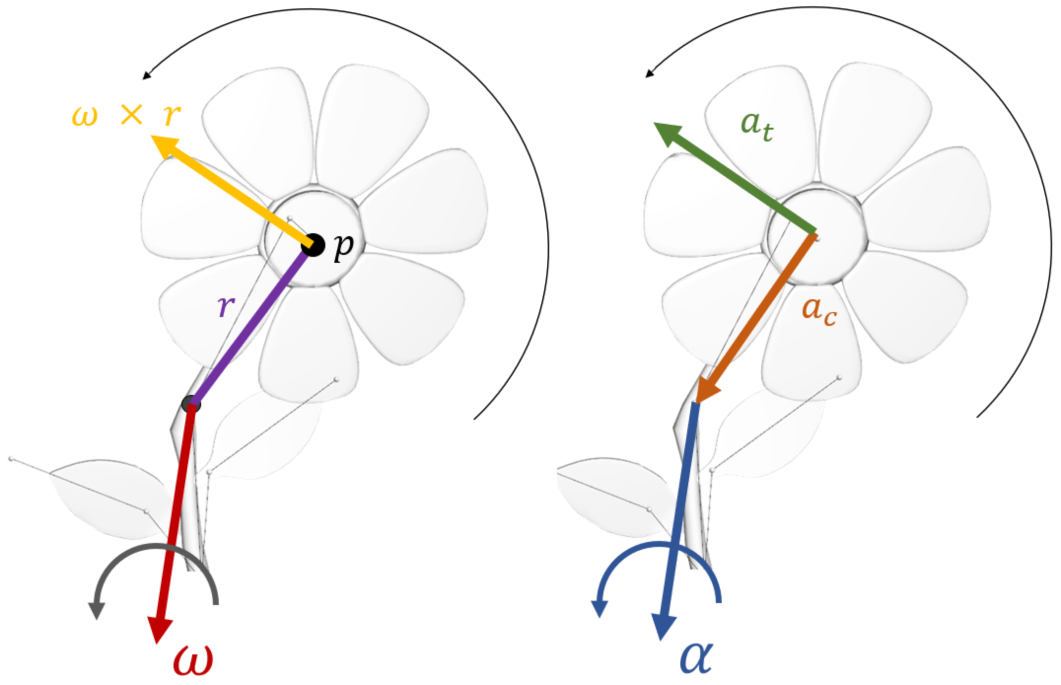

Figure 4.3: Velocity (left) and acceleration (right) terms visualized on rotational motion on a flower.

Within the context of Velocity Skinning, it makes sense to envision this effect as another deformation function ψ^{ft_o} which may be added to the summation.

$$d(p) = \sum_j \tilde{b}_j \psi^{ft_o}$$

This function may again be decomposed into linear and angular functions corresponding to translational and rotational motion.

$$d(p) = \sum_j \tilde{b}_j (\psi^{ft_o}_{tr} + \psi^{ft_o}_{rot}(p))$$

To create the followthrough effect, the target mesh must be deformed beyond the skeleton's rest position, ie, the mesh must deform even after the skeletal animation has come to a stop. If we attempt to create this effect using velocity, we run into a problem; once the skeleton stops moving, both linear velocity u and angular velocity ω are zero, and we no longer have any input signals to work with. This is where we reach the limitations of working solely with velocity. Acceleration, however, provides us with further possibilities, as discussed in section 4.1.

Utilizing these acceleration terms in our deformers, we have $\psi^{ft_o}_{tr}$ as a function of linear acceleration a.

$$\psi^{ft_o}_{tr} = \psi^{ft_o}_{a_j} = -a_j$$

For rotational motion, we have two separate terms to consider: a_t and a_c. We can see from Fig. 4.3 that tangential acceleration a_t provides a vector that is axis aligned with the vertex acceleration $\omega \times r$. This is the same value used in the rotational drag deformer $\psi^{dv}_{\omega_j}$. To recap,

$$\psi^{dv}_{\omega_j}(p) = (R(\theta_\omega) - I)(p - p_j).$$

Here, the ω value is present as part of the rotation matrix R, as the axis of rotation. We have the angle of rotation

$$\theta_\omega = -k_{dv}||v_i||.$$

In the original Velocity Skinning formulation, the vertex velocity v_i is computed as the velocity caused by the rotation of vertex position p about the bone origin p_j, expressed as

$$v_i = \omega \times (p - p_j).$$

Therefore, we have the angle of rotation due to vertex velocity θ_ω defined as

$$\theta_\omega = -k_{dv}||\omega \times (p - p_j)||. \tag{4.3}$$

Substituting α in place of ω, we have angle of rotation due to tangential vertex acceleration

$$\theta_\alpha = -k_{ft}||\alpha \times (p - p_j)||. \tag{4.4}$$

where k_{ft} is a user-controlled scalar value. Notice that this formulation resembles Eq. 4.1. Therefore, we rewrite Eq. 4.4 as

$$\theta_\alpha = -k_{ft}||a_t(\alpha, (p - p_j))||.$$

Fig. 4.4 shows these terms visualized on a rotating flower. Finally, updating the deformation function to reflect these modifications, we have

$$\psi_{rot}^{ft_o} = \psi_{\alpha_j}^{ft_o}(p) = (R(\theta_\alpha) - I)(p - p_j). \tag{4.5}$$

Fig. 4.5 showcases the effect of $\psi_{\alpha_j}^{ft_o}$ applied onto a rotating flower animation.

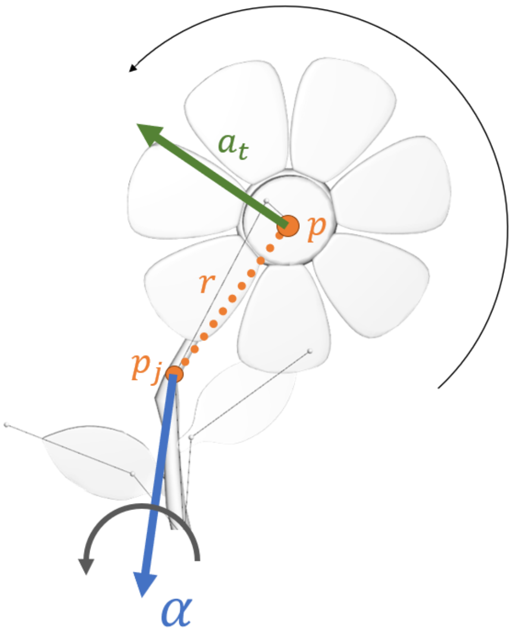

Figure 4.4: Tangential acceleration visualized on a rotating flower.

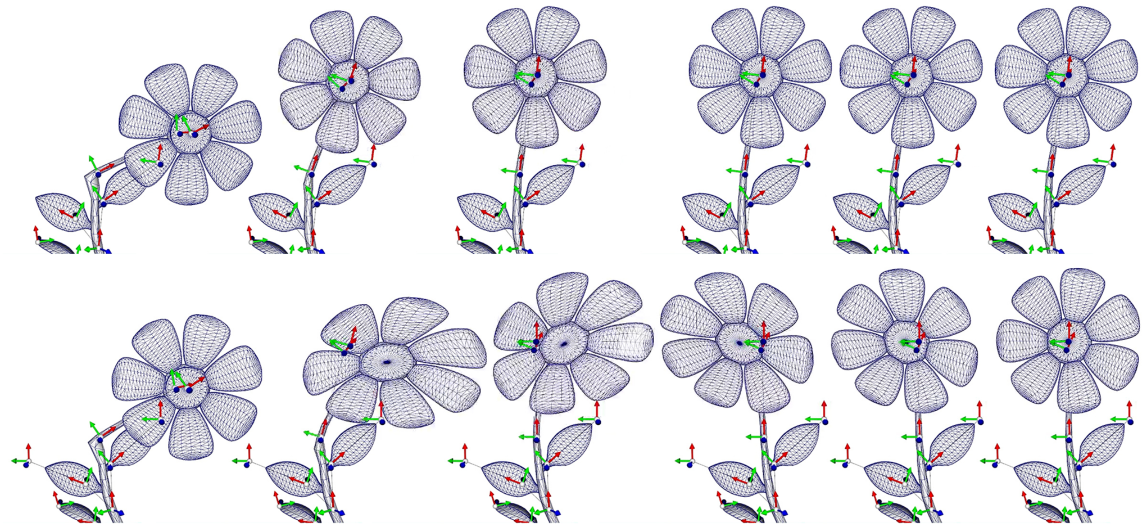

Figure 4.5: A simple rotating flower animation (top) and the results of applying the followthrough deformer from Eq. 4.5 on it (bottom)

Chapter 5

Deformers

5.1 Followthrough

5.1.1 Followthrough Warping

Applying ψ^{ft_o} using the existing system results in both drag deformation (where deformation causes the mesh to "trail behind" the skeleton) as well as followthrough deformation (where deformation causes the mesh to shoot in front of the skeleton.) As both of these deformation effects are controlled by a single acceleration input, modifying the magnitude of acceleration affects the magnitude of both deformations (drag and followthrough). However, it is more desirable to provide artists with independent control of each effect.

Acceleration denotes the rate of change of velocity. When the velocity and acceleration vectors point in the same direction (Fig. 5.1), the magnitude of velocity increases. Within the current context of the followthrough deformer, this configuration of ω and α vectors causes the drag effect. In Fig 4.1, this corresponds the first (positive) α lobe. At this point, both α and ω are positive. When these vectors are pointing away from each other (Fig. 5.1), the deformer generates an effect of the mesh shooting in front of the skeleton, rather than trailing behind it. In Fig 4.1, this corresponds to the second α lobe, where α is negative (deceleration) and ω is positive. This produces followthrough. This is the effect we truly like to capture with the followthrough deformer.

Using the intuition stated above, we see that a comparison between the ω and α vectors provides the information needed to determine whether to engage the drag or followthrough effects.

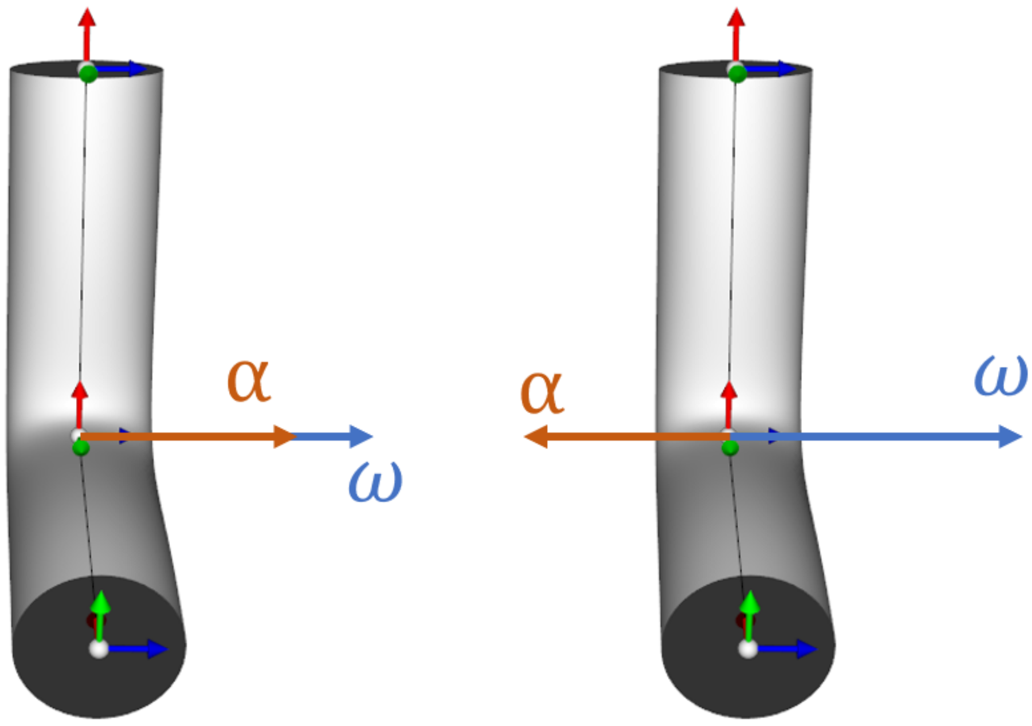

Figure 5.1: ω and α vectors visualized on a bending cylinder in cases where they are aligned (left) and pointing in opposite directions (right)

This is done using a simple dot product $\omega \cdot \alpha$. Computationally, we handle this by scaling the existing ψ^{ft_o} with a function (which we call *warp* function). This function is generated using the dot product $\omega \cdot \alpha$, remapped to fit within the [0, 1] value range. The final deformation function is defined as follows:

$$\psi^{ft}_{\alpha_j}(p) = \psi^{ft_o}_{\alpha_j}(p) f(\omega_j \cdot \alpha_j, limit) \tag{5.1}$$

$$f(\omega \cdot \alpha, limit) = \begin{cases} 0 \text{ if } \omega \cdot \alpha \geq 0, \\ \frac{|\omega \cdot \alpha|}{limit} \text{ if } 0 \geq \omega \cdot \alpha \geq limit, \\ 1 \text{ otherwise} \end{cases}$$

where

$\psi^{ft}_{\alpha_j}$ is the warped rotational followthrough deformer,

$\psi^{ft_o}_{\alpha_j}$ is the original rotational followthrough deformer before warping,

f is the warp function, and

limit is a user-defined limit or threshold representing the transition between the drag and followthrough portion of the input signal.

The effect of the warp function applied onto an input acceleration signal can be seen visualized in Fig. 5.2 (a and b). We can perform a similar warping on translational followthrough, defined as:

$$\psi^{ft}_{a_j} = \psi^{ft_o}_{a_j} f(a_j \cdot v_j, limit), \tag{5.2}$$

where $\psi^{ft}_{a_j}$ is the warped translational followthrough. This gives us a "warped signal" containing just the followthrough deformation (Fig. 5.2b), giving artists the ability to target and independently tweak the followthrough deformation as per their aesthetic requirements.

5.1.2 Tangential Acceleration Drag

Now that we have the capability to separate the drag and followthrough effects independently, it is worth considering the nature of drag as modelled by acceleration, as opposed to velocity. These two functions result in different deformations. When we model drag using velocity, we have

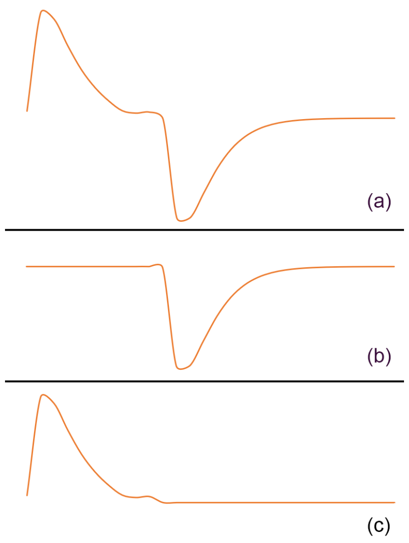

Figure 5.2: Input acceleration signal (a), warp function applied to extract followthrough (b), warp function used to extract acceleration-based drag (c)

a certain smooth deformation over time. Computing acceleration using finite differences, however, leads to increased sensitivity to small variations and noise. When we model the drag effect using acceleration, the deformation captures and reflects these smaller changes (Fig. 5.3). Artistically, both of these approaches have merit. Velocity-based drag may be preferred to mimic the effect of moving through a medium with higher friction. Acceleration-based drag may be preferred when more subtleties are desired in the deformation. We may repurpose the warp function to create an acceleration-based drag deformer:

$$\psi^{da}_{\alpha_j}(p) = \psi^{ft_o}_{\alpha_j}(p)[1 - f(\omega_j \cdot \alpha_j, limit)]$$

$$\psi^{da}_{a_j} = \psi^{ft_o}_{a_j}[1 - f(a_j \cdot v_j, limit)] \tag{5.3}$$

where

$\psi^{da}_{\alpha_j}$ is the rotational drag deformer modeled using angular acceleration α, and

$\psi^{da}_{a_j}$ is the translational drag deformer modeled using linear acceleration a.

We may modify the k_{ft} term used in ψ^{ft_o} independently for ψ^{ft} and ψ^{da} to allow users to independently control the intensities of followthrough and acceleration-based drag deformations

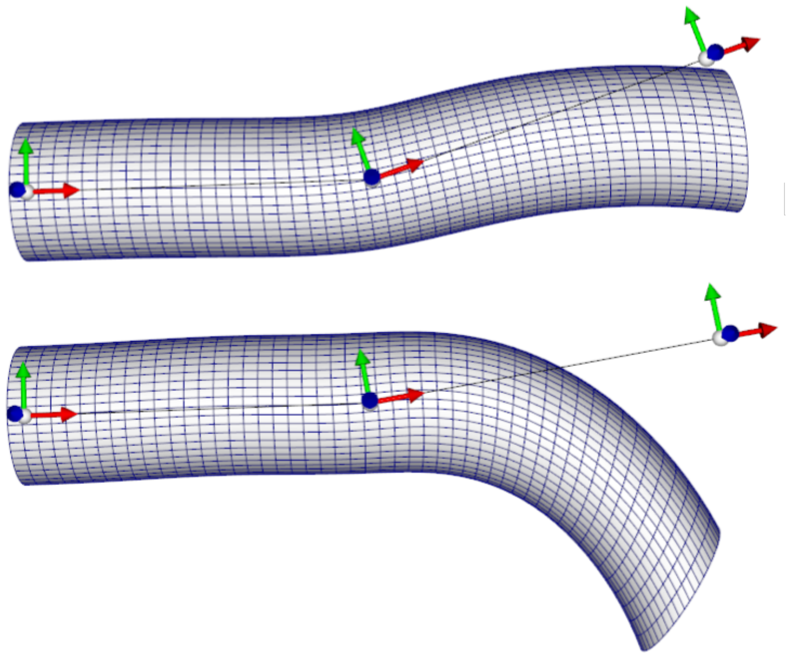

Figure 5.3: ω (top) and α (bottom) driven drag effect at moments where the animation changes directions.

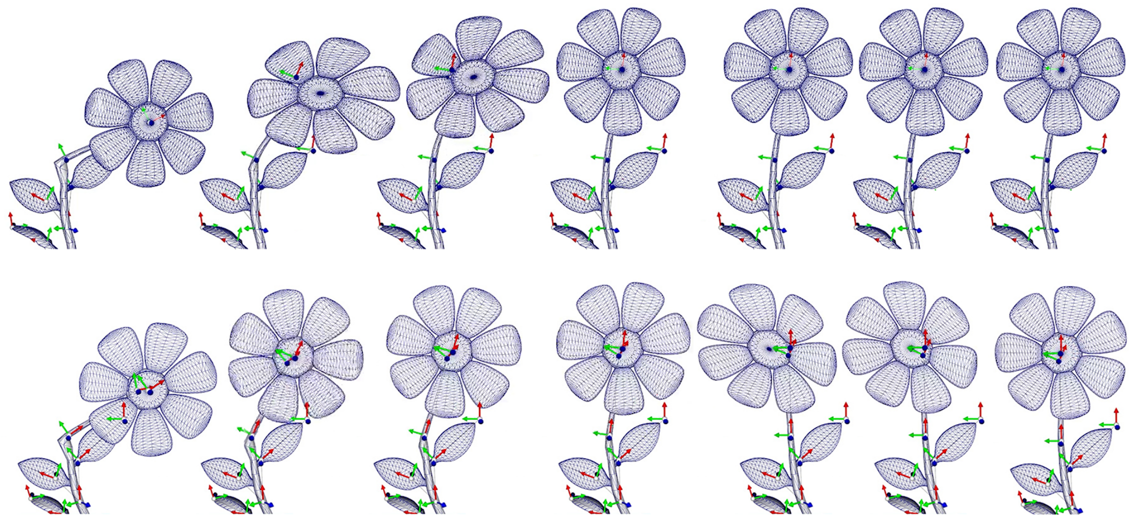

Figure 5.4: Results of deforming the source flower animation in Fig. 4.5 using acceleration-based drag ψ^{da} (top) and warped followthrough ψ^{ft} (bottom)

respectively. Fig. 5.2 (a and c) show the effects of the warp function used to extract acceleration-based drag from an input acceleration signal. Fig. 5.4 shows the results of applying the warp function on the previously seen flower animation (Fig. 4.5).

Working with acceleration-based drag may occasionally lead to an interesting intricacy. Consider a scenario in which the source animation has a constant velocity. The resulting derivative of velocity in this scenario would be zero. In other words, there would be no acceleration, and by extension, no deformation. This is simply a by-product of how acceleration works. Of course, depending on the context of the animation, this may be preferred to having constant deformation. For example, if the source animation switches between several different constant velocities, then applying acceleration-based drag would cause deformation only at the points of change. Depending on the context, this may be a desirable effect. Again, it is up to the artist to decide where and how they wish to use these effects.

5.2 Centripetal Stretch

The centripetal acceleration of a rigid body:

- pulls the body radially inward toward the center of rotation,

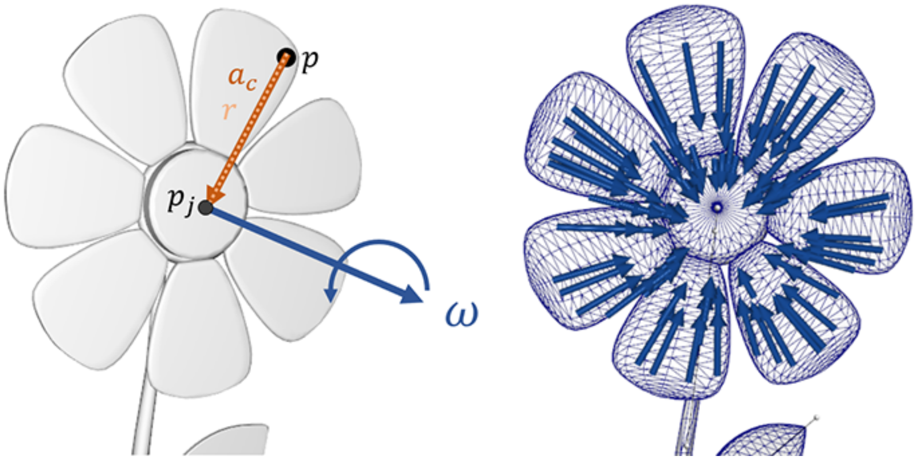

Figure 5.5: Centripetal acceleration a_c of a vertex acts radially inward toward the point of rotation

- has a magnitude proportional to the radius of rotation.

When observing how chefs make pizza, we see that after kneading and pressing the dough, they proceed to toss it in the air with a spin of their wrists. This spin creates creates a force that stretches the dough farther out. We can clearly see the relevance of centripetal acceleration in this process. This is what inspired the creation of the *Centripetal Stretch* deformer.

5.2.1 Computation

To model this effect, we need a deformer function ψ^{cs} that computes the deformation for any given vertex on the mesh. By simple adapting Eq. 4.2, we have

$$\psi^{cs}(p) = -k_{cs} a_c(\omega_j, p - p_j)$$

where

ω_j is the angular velocity of the rotating joint,

p_j is the joint origin, and

k_{cs} is a user defined effect scale.

Recall that the magnitude of centripetal acceleration acts inwards toward the point of rotation, as visualized in Fig 5.5. Our deformation needs to push vertices outward, away from the center. Hence, the output of the deformer is negated to reverse the direction.

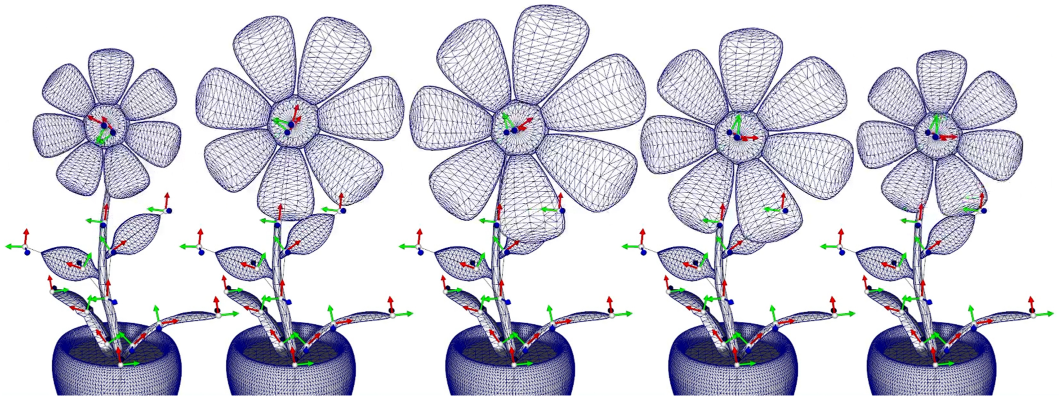
Figure 5.6: Centripetal stretch effect applied onto an animation of a flower twisting its head

This creates an effect that translates vertices outward away from the point of rotation. As the magnitude of acceleration increases linearly with distance, the vertices farther away from the center of rotation experience larger deformation than those closer to the center. To exaggerate this effect, we replace the coefficient k_{cs} with a function $k(x)$ to scale the deformation effect with respect to p. Applying quadratic mapping, we have:

$$\psi^{cs}(p) = -k(p - p_j)a_c(\omega_j, p - p_j)$$

$$k(x) = k_{cs}||x||^2$$

The effects of the centripetal stretch deformer can be seen applied to a twisting flower animation in Fig 5.6.

5.3 Centripetal Lift

Consider the motion of a twirling cloth, such as a dancer's skirt. As the dancer twirls, their skirt gently lifts. The same centripetal forces that caused the pizza dough to expand also cause this effect on the cloth. Thus far, we have exploited the concepts of angular acceleration, tangential acceleration, and centripetal acceleration. However, to model this effect using the acceleration skinning framework, we explore another option which I call “centripetal lift”.

5.3.1 Computation

Consider the eventual goal described above: attempting to make a dress lift as it twirls. Assume we have cylindrical cone representing a simplified version of the skirt. Assume this cone twists about an axis aligned to its height. The drag deformer creates a natural twirl effect, putting us half way towards achieving our goal. Now we need a deformer to cause the dress to lift. Consider Fig 5.7. From this diagram, we can see that the vertex normals of the cone are somewhat aligned with the direction in which we would like to displace these vertices to create a lifting effect. However, blindly translating these vertices along the normal would not serve any purpose. To achieve our desired effect, we need the ends of the cone to lift (or deform) more than the tip of the cone. Recall that our drag deformer also exhibits this same behavior. Therefore, we can use the drag deformer to control the magnitude of the lift. Based on this intuition, we have the following equation:

$$\psi^{cl} = k_{cl}||\psi^{da}||\hat{n_i}$$

where k_{cl} is a scalar coefficient to control the deformation intensity.

We now have a basic deformation function that works. However, we are only considering the influence of drag. Ideally, the effects of other deformers should contribute towards the lifting of the cone as well. To make this more generic, consider replacing ψ^{da} with d_1 which is a linear sum of all deformation functions supported by the acceleration skinning system.

$$d_1 = \psi^{da} + \psi^{sv} + \psi^{ft} + \psi^{cs}$$

$$\psi^{cl} = k_{cl}||d_{1_i}||\hat{n_i}$$

The issue with this equation is that while it considers the magnitude of all the deformers in d_1, it does not take into account the directional effect of these vectors. This is rectified by introducing a dot product to take this alignment of vectors into account as follows:

$$\psi^{cl} = k_{cl}(d_{1_i} \cdot \hat{n_i})\hat{n_i}$$

Directly injecting this deformer function into the existing acceleration skinning summation causes a recursion issue; we require the summation of all deformers as an input to ψ^{cl}, but must ensure

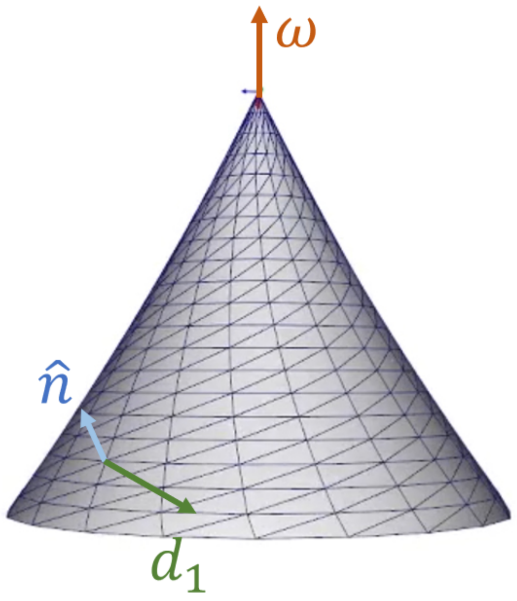

Figure 5.7: Vectors visualized on the surface of a conical mesh to compute centripetal lift deformation.

that ψ^{cl} is not passed as an input to itself as part of the summation. This can be accommodated by revising the acceleration skinning formulation as two separate passes:

$$d_1(p) = \psi^{da}(p) + \psi^{sv}(p) + \psi^{ft}(p) + \psi^{cs}(p)$$

$$\psi^{cl}(p) = k_{cl}(d_{1_i}(p) \cdot \hat{n_i})\hat{n_i}$$

$$d(p) = \sum_j \tilde{b}_j(d_1(p) + \psi^{cl}(p))$$

Fig 5.8 and 5.9 showcase the effects of the centripetal lift deformer applied on top of drag (5.8) or drag with centripetal stretch (5.9).

5.3.2 Limitations

The centripetal lift deformer, in its current state, has the following limitations:

- It can only be applied on a cylindrical cone, cylinder, or frustrum. Applying the deformer to other shapes creates unappealing deformation due to the change in flow of vertex normals.
- The base(s) of this base geometry must not be closed. Applying the deformer on closed meshes

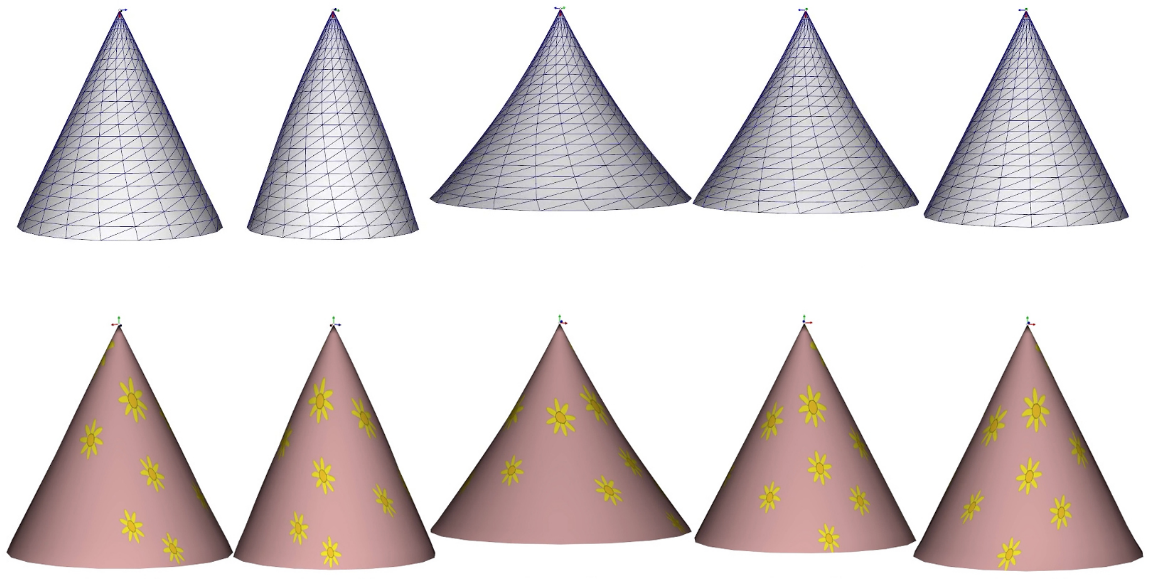

Figure 5.8: Effects off ψ^{da} and ψ^{cl} deformers on a simple twist animation. Wireframe view (top), textured view (bottom).

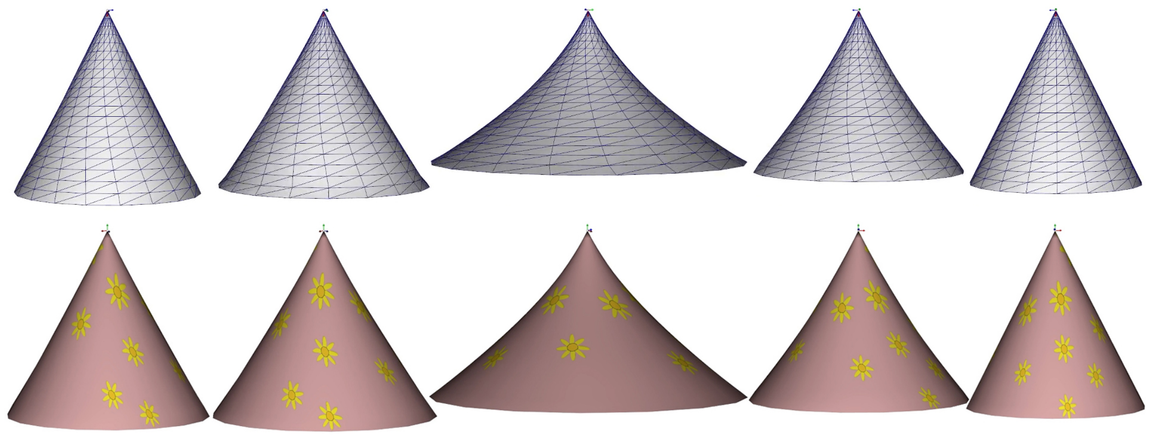

Figure 5.9: Effects off ψ^{da}, ψ^{cs}, and ψ^{cl} deformers on a simple twist animation. Wireframe view (top), textured view (bottom).

causes abnormal swelling of the tips of the mesh, since all of the vertex normals point radially outward. Results from this experiment can be seen in Fig 5.10.

- All of the selected geometry listed above lack a base; they have hole(s) in the mesh. This does not comply to industry standard practices of using "thick-walled" geometry. If you require a mesh with a hole, it is common practice to extrude the relevant polygonal faces on the mesh to give it some thickness. As we saw, however, adding thickness to our base geometry causes issues with the deformation results

Theoretically, these limitations may be addressed with a workaround: applying the deformation onto a procedurally generated $deformation\ frustrum$ and mapping the resulting displacement onto a given a target mesh. Assume that the $deformation\ frustrum$ is procedurally generated with the following parameters:

- The user selects a target joint or joint chain.
- The height of the frustrum matches the height of the target joints (or the vector connecting the end points of a targeted joint chain).
- The radii of the bases of the frustrum match the radii of the cross section of the target mesh at the ends of the targeted join or joint chain.
- The frustrum is positioned in the same space as the targeted mesh such that the two meshes overlap each other.

We generate a deformation frustrum and apply a mapping scheme to transfer the vertex displacement from one mesh onto the other. A number of different mapping schemes may be used to accomplish this. For instance, we may ensure that the vertex count on the deformation frustrum is equal to the number of vertices on the target mesh, and then apply a bijection function to perform 1:1 correspondence mapping between the two. Each vertex on the deformation mesh is responsible for deforming a unique vertex on the target mesh. If desired, more complex mapping involving Nearest Point Transform or Iterative Closest Point Transform [6] [26] may be used.

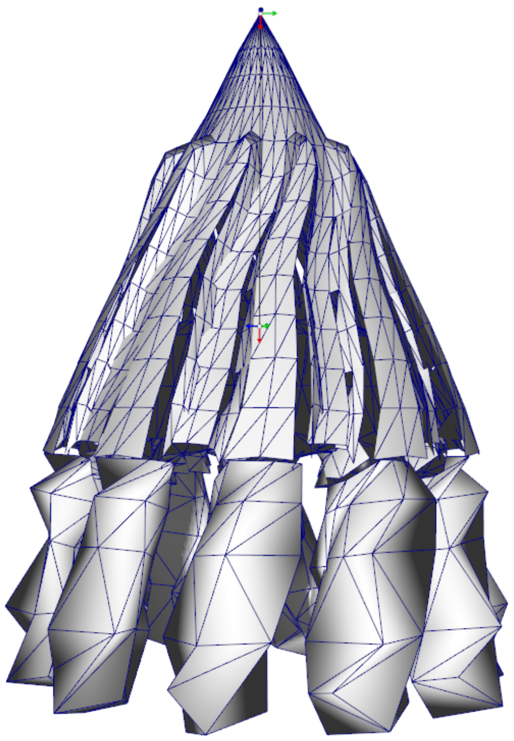
Figure 5.10: Effects off of the centripetal lift deformer applied onto non-conical geometry

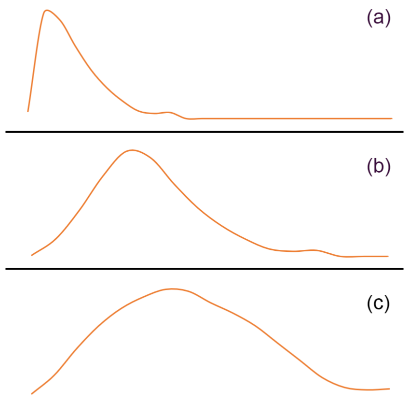

Figure 5.11: A given input signal processed by increasingly larger time filters from (a) through (c).

5.4 Effect-Specific Time Filter

For all of the examples showcased thus far, the input signals (u, a, ω, α etc.) have been filtered over time. Rather than globally applying a static time filter on these terms, we may apply temporal filters independently for each effect. In other words, each deformer function has it's own unique input for the velocity and acceleration terms (ω_{dv}, ω_{sv}, ω_{cs}, etc). The size of the time filter is a user controllable value, isolated to each individual deformer. Increasing the size of the filter leads to slower deformation, as more frames contribute to the filtered value. Conversely, reducing the filter size yields snappier deformation effects. The changes in signal are visualized in Fig. 5.11. A comparison between using a smaller and larger time filter on a flower animation clip is presented in Fig. 5.12.

Let us explore a practical example that makes use of this technique. The canonical approach to combining deformation functions thus far has been to compute a linear sum of the effects. This, however, leads to certain specific issues. The centripetal stretch deformer is active when there is centripetal acceleration, and the followthrough deformer is active when there is tangential deceleration. Consider the state of these quantities immediately after the skeleton of an animated character has come to rest. Depending on how fast the skeleton was moving and how fast it came to rest, some tangential deceleration may be present, but there will not be any centripetal acceleration. a_c is always zero when the skeleton is at rest. Therefore, in such circumstances, ψ^{cs} will hit zero deformation before ψ^{ft} starts to act. This is not aesthetically pleasing (see Fig. 5.13). Effect-specific time filtering may be used in this scenario to "blend" the two deformation effects together. In Fig 5.14,

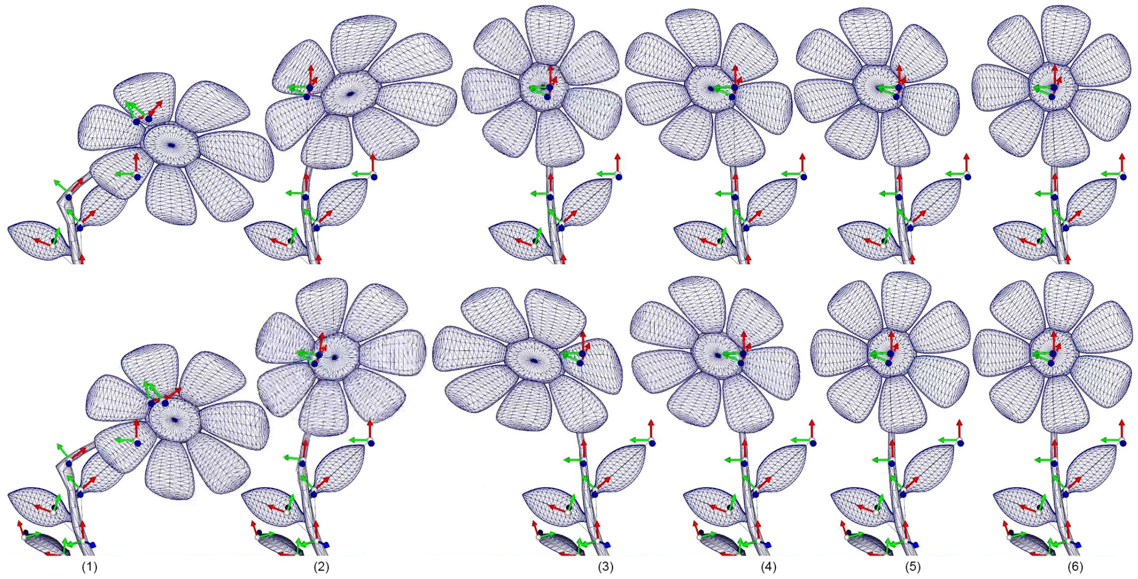

Figure 5.12: Comparison between the use of a large time filter (top) and a small time filter (bottom) applied on the acceleration-based drag and followthrough deformers. Frames (2), (3), and (5) emphasize the differences between the two filters, showing that small filters increase deformer reactivity.

we see the results of the same previous example, but with artist-tuned time filters. This result is obtained by using:

- Slower centripetal stretch (larger time filter)
- Faster followthrough effect (smaller time filter)

Exploiting this new approach provides the following benefits:

- We enhance the artists' control of the effects by allowing them to further tune the results.
- The deformation response can be modified independently per effect.
- We are able to retain the system's existing approach to combining effects by computing their linear sum.

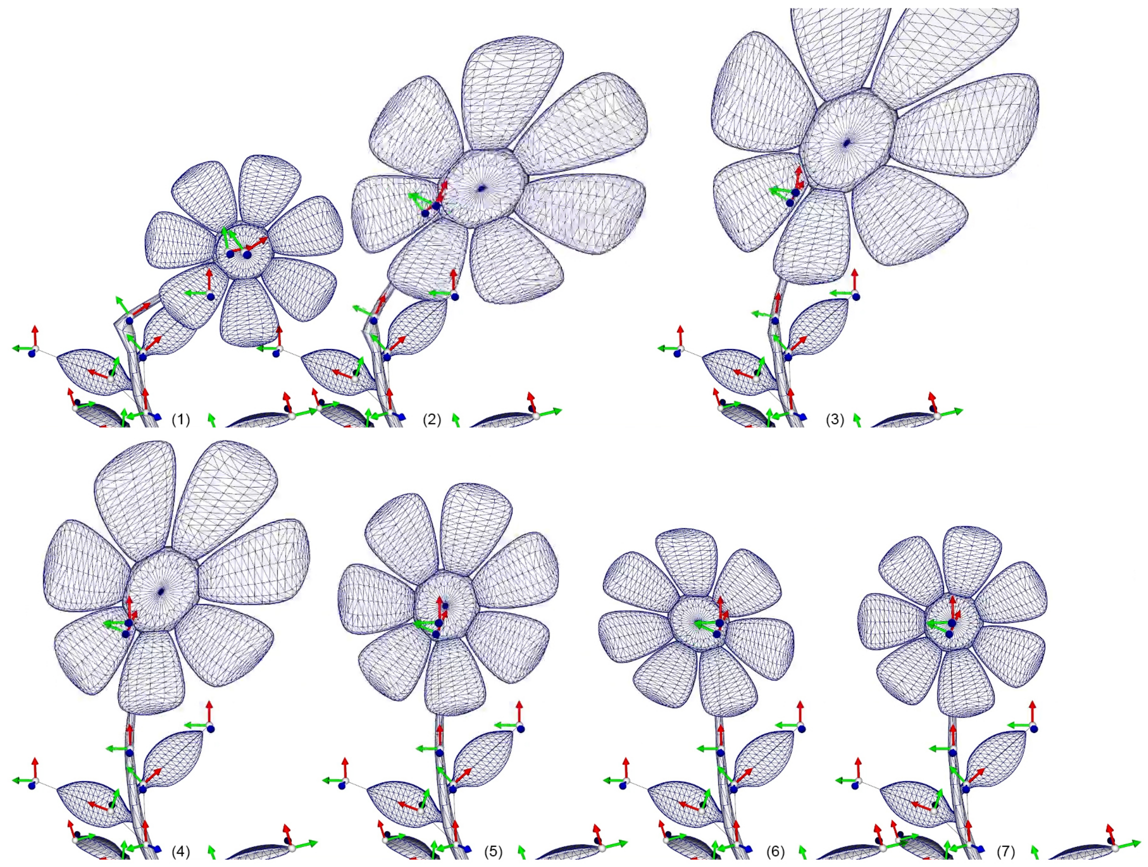

Figure 5.13: Result of combining deformers using a linear sum. Follow-through and Centripetal Stretch effects are “mutually exclusive” in this example; they do not overlap. Centripetal stretch deformer is active from images (1) to (5), followthrough deformer is active from (5) to (7).

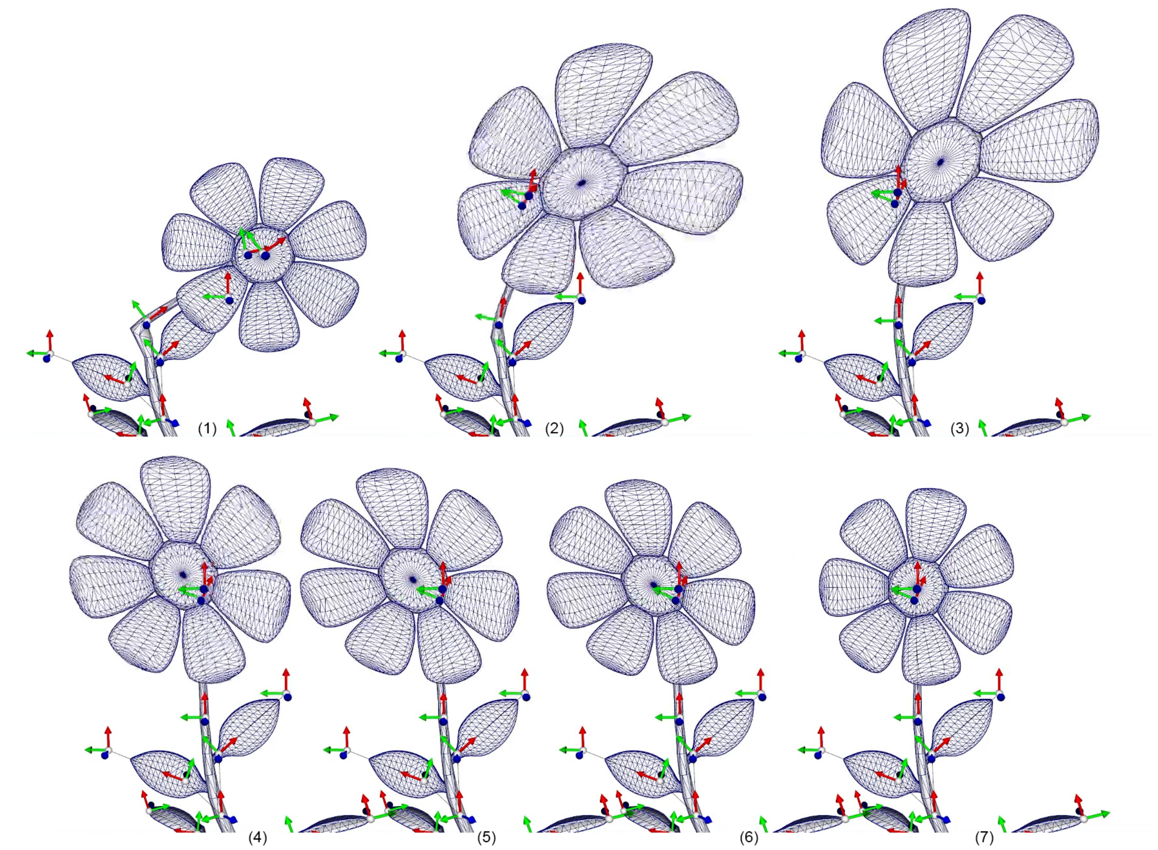

Figure 5.14: Results from artist-tuned time filters. Comparing these results with fig. 5.13, we see that the centripetal stretch and followthrough effects are now blended together. This change is particularly visible on images (3), (4) and (5).

Chapter 6

Production Evaluation

Once we had a stable system for velocity skinning, I conducted a study to evaluate the usability of this tool in a production environment. Volunteer animators provided character animation as input, and I assumed the role of an effects artist to test out this system. This study was conducted before extending velocity skinning to include acceleration, so only a subset of the tools and deforms discussed were available for use in the final result.

6.1 Procedure

Our implementation of the velocity skinning system was originally built to import rigged characters from custom ASCII data files. These rigs were typically created in Blender and exported to Collada format. These Collada files were then parsed to extract relevant data and stored in the custom data format for injection into the system. A rudimentary real-time animation functionality was supported allowing the user to move and rotate joints on the skeleton and capture these results as skeletal animation. However, professionally trained 3D artists generally prefer to work in Autodesk Maya over Blender. Maya is also the industry standard for rigging and animation work. Hence, I modified this pipeline to support integration with Maya. I wrote a set of python scripts in Maya to parse a targeted character rig and export its data to our proprietary format. I later extended this functionality to support character animation as well, allowing us to work with higher quality animation input.

The models used in this study were obtained from various royalty free sources online. Clem-

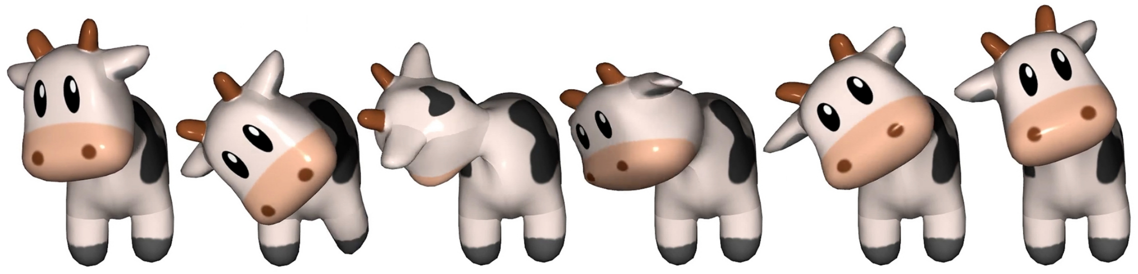

Figure 6.1: Deformed cow animation sequence from section 6.2.1

son student (now instructor) Rodney Costa volunteered to rig these characters to be more animator friendly. We had three students from Clemson University's Digital Production Arts MFA program graciously volunteer to be animators for this study. They all unanimously agreed to create character animation as per our specifications to provide us with higher quality artist input. Their experience levels with character animation varied; some were beginners while others had prior industry experience. Unfortunately, they were not as inclined to step outside of Maya. Finally, they chose the rigs they wished to use. They animated their characters. I then exported this animation and then applied and tweaked the deformations using our velocity skinning system. More detailed walk-throughs of my process and approach to applying these deformations are available as part of supplemental material of our Eurographics 2021 publication [32].

6.2 Results

I used our system to perform deformation on four different animation inputs.

6.2.1 Cow Animation 1

I personally animated this input using the realtime user capture tools available in our velocity skinning implementation. As a result, this animation lacks the subtleties achievable using more robust animation tools. Applying deformation on this animation was a very easy and straightforward process. The results obtained out-of-the box were very fluid. Minimal amounts of tweaking the input parameters quickly yielded more tame results. I turned down the squash and drag intensities to make the effects more subtle. I then proceeded to modify the velocity paint weights on the mesh, with the objective of emphasizing more deformation on the cows head, particularly on

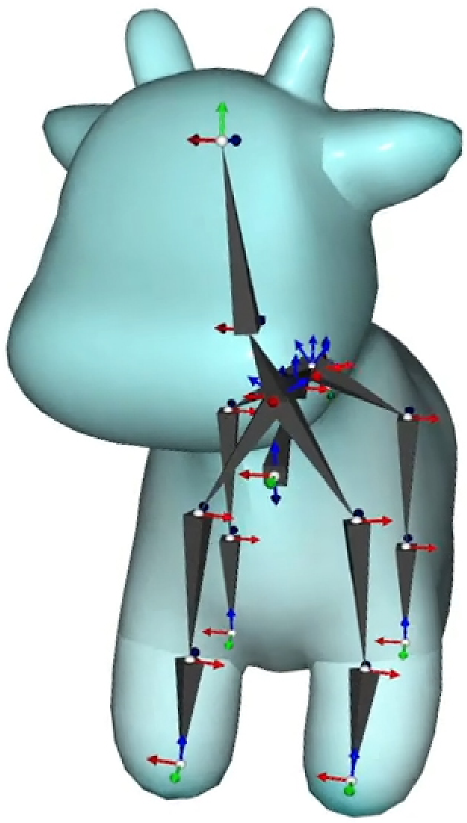

Figure 6.2: Final distribution of paint weights on the cow model used in 6.2.1

the tips of the horns and ears. I also added a bit more weight to the bottom of cows front feet. This is because, in the source animation, the cow kicks his front left foot a bit, and I wanted to exaggerate this movement using the drag deformer. The final distribution of paint weights can be seen in Fig. 6.2. The final results are shown in Fig 6.1.

6.2.2 Dragon

A dragon fly cycle was animated by a volunteer with professional animation experience. I approached the effects on this animation by taking a minute to observe the characteristics of the cycle, and then turned on the deformers at default settings to see what we get. The immediate results were far too exaggerated for my taste, but my goal was to identify which parts of the deformation had the potential to be used in the final results. At this stage, I liked the deformation seen on the hands

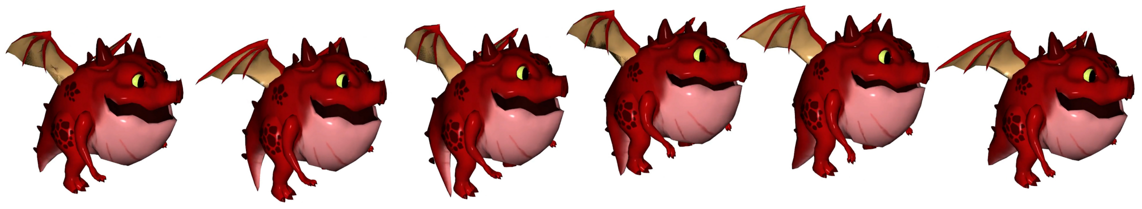
Figure 6.3: Deformed dragon animation sequence from section 6.2.2.

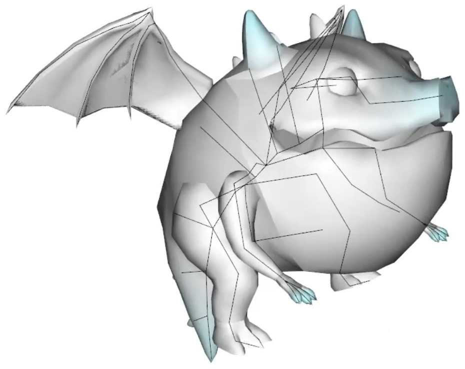
Figure 6.4: Final distribution of paint weights on the dragon model used in section 6.2.2

and horns of the dragon. I then painted weights for the character, using this intuition to guide me. I chose to use extremely sparse weights, isolating the tips of the horns, fingers, snout, and tail. The final paint weights can be seen in Fig. 6.4. After painting the weights, I turned off all of the deformers. I wanted the deformation to be subtle, so starting with all the effect intensities/magnitudes scaled down to zero felt like a logical way to approach this clip. I slowly introduced a bit of drag, and once I was happy, started adding in squash. Combining both of these effects together on this particular clip added bounciness to the motion. Finally, I added some translation drag to introduce some overlap in relation to the rotational deformer. In conclusion, majority of my time on this clip was invested in coming up with an end goal; deciding how to deform the character. After deciding that, the remainder of my time was spent on painting the vertex weights. Out of the entire process, tweaking the effect intensities took the least amount of item. The final results are shown in Fig 6.3.

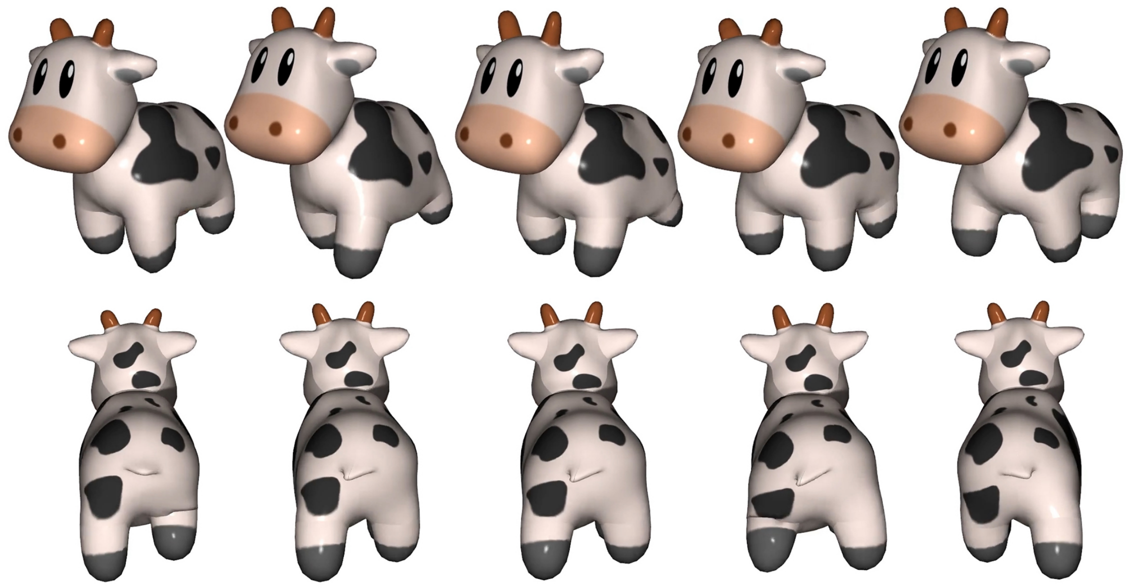

Figure 6.5: Deformed cow walk cycle from section 6.2.3. Subtle deformation is seen on the horns and ears. The hips have a natural side-to-side sway.

6.2.3 Cow Animation 2

One of the volunteers chose to animate a walk cycle on the cow rig. It was a very plain walk cycle, with a bit of bobbling on the head to add some personality. I began by adding zero-ing the intensity of all deformers and gently adding in some drag rotation. The cow's head movement caused the horns to bobble back and forth, adding a lot of contrast to the motion. Introducing drag translation capitalized on the cow's side to side motion, and created an interesting effect. Personally, I was very captivated by how it delayed the cow's tail. Once again, after I reached a point where I was satisfied with the overall deformation and had a fairly clear idea of what I wanted the final result to look like, I started to paint vertex weights. After completing a first pass on the weights, I had a decent amount of deformation on the cow's head, and toned down the effect on the body. (Fig. 6.6) However, I had lost the effect on the tail that captivated me at the start of the process. While refining the painting and redoing the tail, I had the idea to use just a spot of negative weights at the tip of the tail (Fig. 6.7). My goal was to make it look like the cow was intentionally flicking its tail while walking, rather than having it simply follow the primary action. In my personal opinion, this added more character to the animation. At this point, I was mostly happy with the results, but the cow's legs were flailing around a lot. I wanted to completely turn off all deformation on

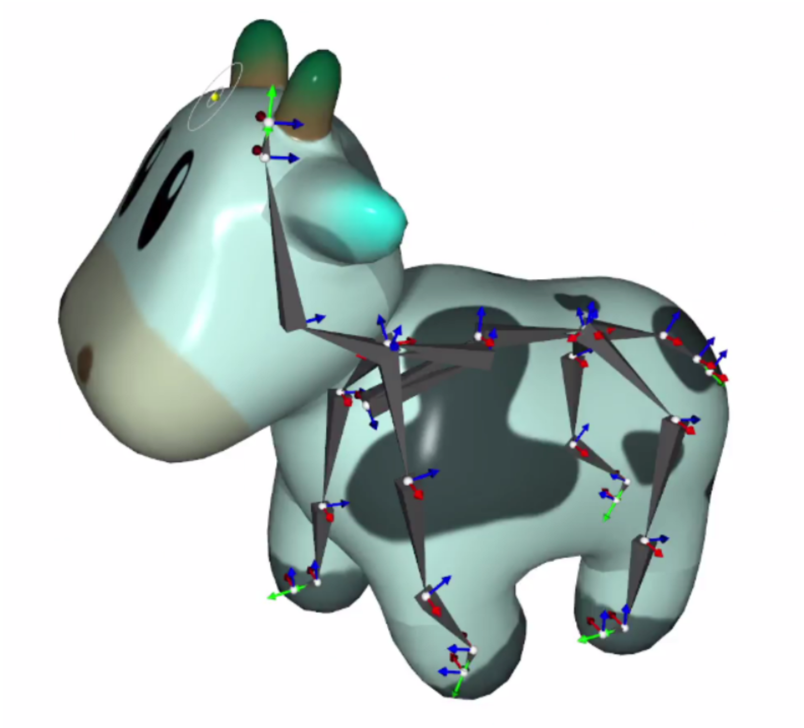

Figure 6.6: First pass of weight painting on the cow model used in section 6.2.3.

these parts. Rather than adjust the paint weights again, I decided to modify the local deformation controls to zero out the drag intensity for the joints on the legs. Re-evaluating my situation at that point, I realized that I had gone overboard with the deformation on the horns and ears. They were too exaggerated for my taste. I revisited the paint weights and turned it down until I was satisfied with the results (Fig. 6.7). In a short duration of time, I was able to take a simple walk cycle and push it artistically further. The final results are shown in Fig 6.5.

6.2.4 Flower

Our final volunteer decided to animate a flower. They were not as experienced in animation and were also unable to spend as much time on the animation compared to other volunteers. As a result, the animation that they shared with me was still a work in progress. Rather than continue their work and cleaning up the animation in Maya, I decided to throw it into Velocity Skinning in its current state. This was an interesting experiment to see what can be done with input that was incomplete. The movements were much broader in this clip compared to the others I worked

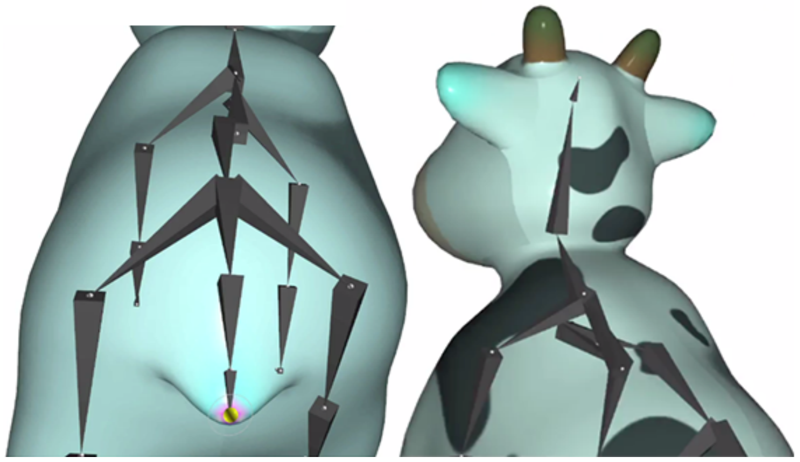

Figure 6.7: Negative paint weights used on cow tail (left); Refined paint weights on the horns and ears (right).

on. I decided to use Velocity Skinning on this one to get a better "preview" of what the animation would look like when it was completed and more polished. As the motion was so broad, applying deformers onto the clip caused extremely exaggerated effects, and in some cases, went as far as breaking the mesh in certain parts. To fix this, I utilized the *deformation visualizer* present in the Velocity Skinning toolkit [32]. This allowed me to freeze a static pose from the deformation and study what it looks like from various angles. I decided to fine tune the paint weights on the mesh in this static pose. This allowed me to clean up any parts of the mesh that were breaking due to excessive deformation. By repeating this process iteratively for all problematic poses, I was able to create paint weights that prevented any unwanted deformations. Fig. 6.9 portrays this process of identifying and tweaking the issue. For the final effect, I decided to remove any deformation from the flower pot as this was hopping around the scene. To go hand in hand with that, I also removed deformations from the base of the stem. The ends of the leaves and flower petals were painted to have the most deformation. Even in such early stages of animation, Velocity Skinning was able to add fluidity to the given movement. This serves as an example of how this technique can also be used for pre-visualization. See Fig. 6.8 for final deformation results.

After completing this study, the final results were shared with the volunteer artists. They were mostly satisfied and enthusiastic. One became even more inspired and requested permission to use this technique in a short film.

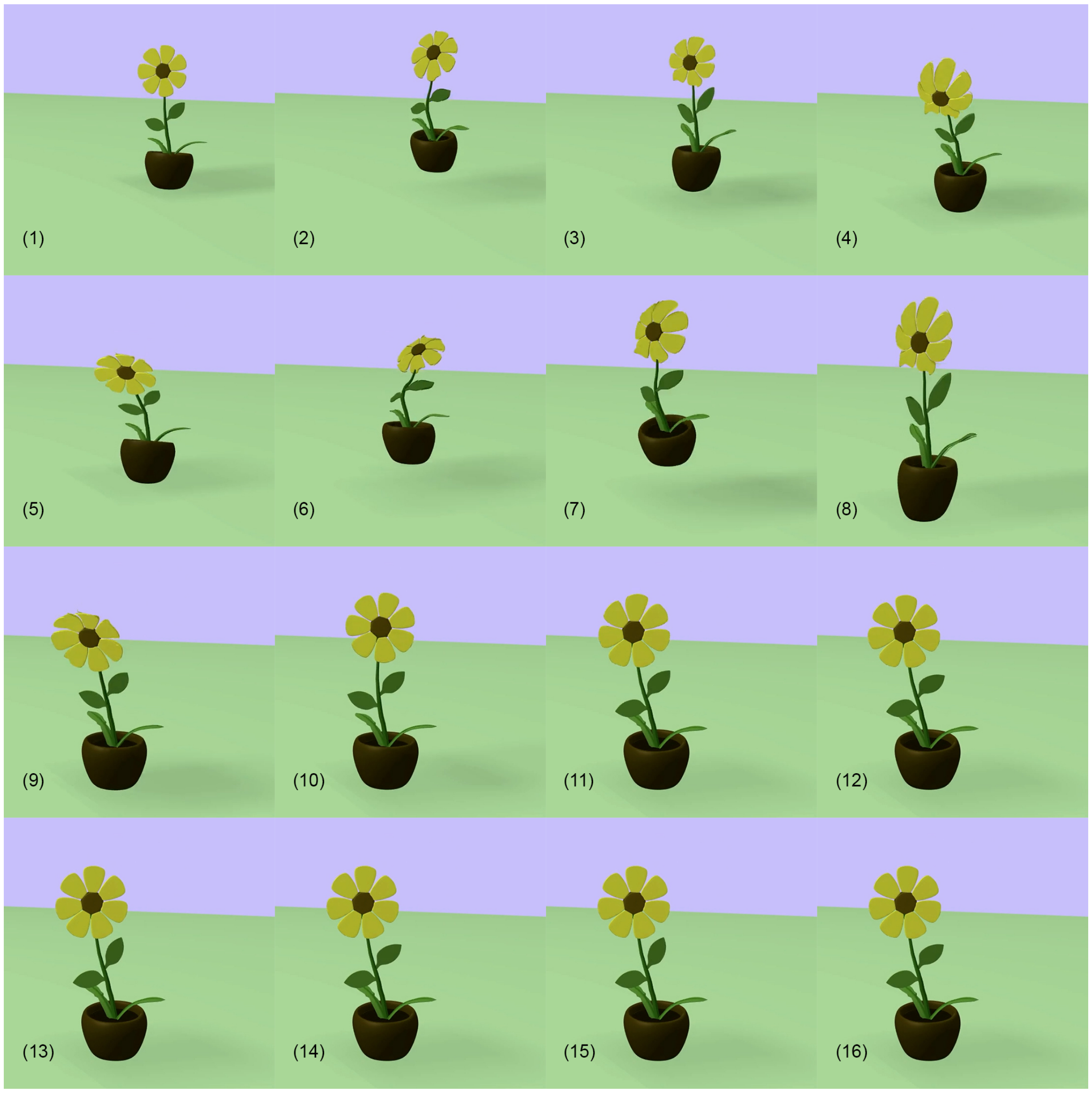

Figure 6.8: Deformed flower animation from section 6.2.4.

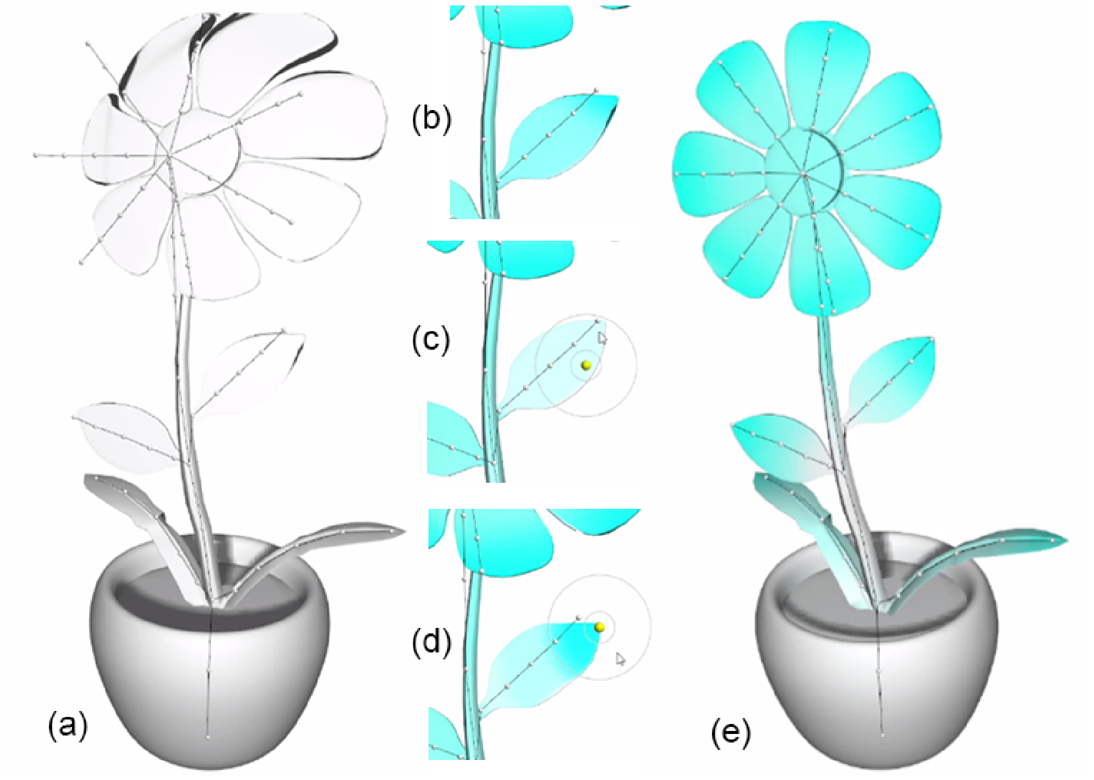

Figure 6.9: Process of cleaning up undesired deformation on problematic poses using per-vertex paint weights. Problematic pose (a), cleanup and deformation fine-tuning process by painting weights (b), (c), (d), final paint weights (e)

Chapter 7

Conclusions and Discussion

7.1 Summary

In this thesis, I introduced Acceleration Skinning as an extension to the recent Velocity Skinning technique by considering the effects of skeletal acceleration. The general idea is to utilize acceleration terms to further expand the gamut of deformation effects available through this skinning system. I showcased some of the possibilities by creating three new deformation functions and tested them through a custom C++ implementation. The refinement of artist control over the deformation through the use of twist-bend decomposition and local joint controls was demonstrated. The inclusion of effect-specific time filtering and its relevance to enhancing artist control was also explored.

A production evaluation of these deformation effects was also conducted, with very promising results. The volunteers were hesitant to use our custom C++ application; they preferred to work in Maya as much as possible. Due to this constraint, they contributed in the role of animators, and I assumed the role of an effects artist, adding secondary motion to their animation. After completing my work, I shared the results with the volunteer animators. They were mostly satisfied and enthusiastic. One became even more inspired and requested permission to use this technique in a 3D animated short film.

In summary:

- Acceleration skinning is a generic framework with an expandable collection of effects.

- Its goal is to use kinematic skeletal motion to create aesthetically pleasing deformation through skinning.
- This method is fast, easy to compute, and works "out of the box" on standard skeletal rigs.
- This method is not intended as a replacement to dynamic simulations, but rather an artist-driven tool to create stylized secondary motion.
- A case study suggested that the technique could be used in a production environment.

7.2 Limitations

- Like Velocity Skinning, Acceleration Skinning is not physically accurate. This method does not track the history of motion over time steps, or consider the effects of mass on deformation.
- The method requires a skeletal rig to work. It does not support rigs which use blend shapes. All of the kinematic terms used as input to the deformation functions are computed based on the rotational and translational motion of the skeletal joints. If there is no skeleton, the deformation functions will not have any input to operate with. Hypothetically, a modification to the method that uses the kinetic motion of individual vertices, rather then the skeletal hierarchy may be conceived. Such an extension may allow support for blend shapes. However, in its current state, this method does not work with non-skeletal rigs.
- Any motion applied to a joint will only cause deformation to propagate down the skeletal hierarchy. The deformation will not propagate up the skeleton. This limitation may hypothetically be circumvented by dynamically modifying the skeletal hierarchy and considering a single targetted joint as the root. However, as of now, this possibility has not been explored. Hence, this method fails to model contact effects between the rigged character and other assets. This is one of the situations where dynamic simulation approaches create far more accurate results.
- Exaggerated motion may lead to self-penetration of different parts of the mesh. As we do not consider the relative position of vertices, we have no mechanism of tracking self-intersections (Fig. 7.1).
- Experiments revealed that rotating successive joints in opposite directions may lead to unappealing scaling deformation. This is due to our approach of combining the contribution of each

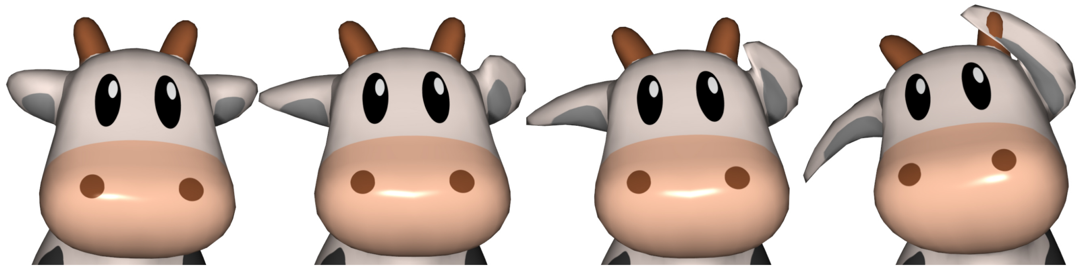

Figure 7.1: Fast skeletal motion or exaggerated deformation my lead to self-intersection on parts of the mesh [32]

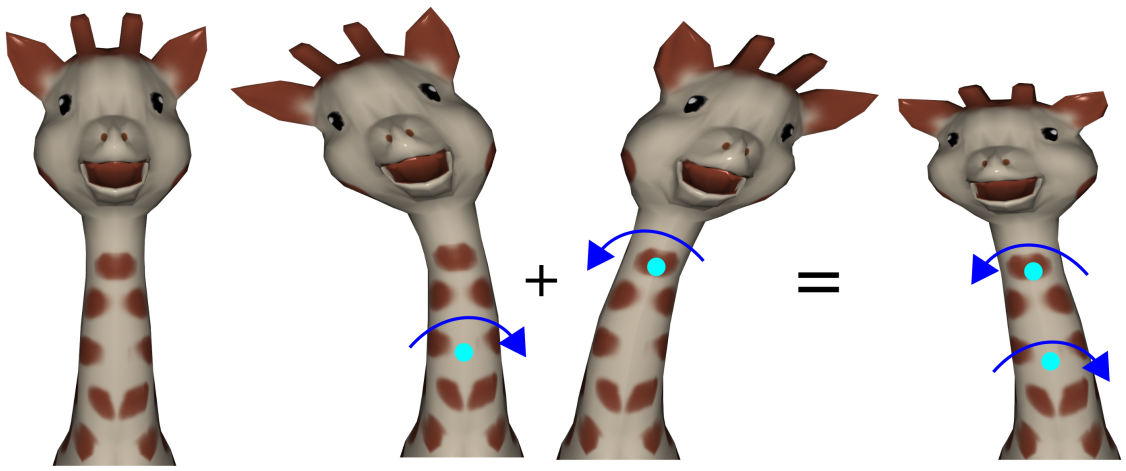

Figure 7.2: Rotation of successive joints in opposite directions may cause unnatural scaling artifacts [32]

joint motion by using a direct sum (Fig. 7.2).

- The proposed centripetal lift deformer only creates appealing deformations on cones, conical cylinders, and frustrums. Theoretical approaches to resolving this issue through the use of mapping systems are discussed in Section 5.3.2.

7.3 Future Extensions

While we chose to create effects that hearken back to traditional animation, we need not limit ourselves to this restriction. The existing system can be expanded to create any effect. The desired effect must be mathematically expressed using the kinematic motion of skeletal joints. Then, the proposed framework may be used to model the desired deformation as a function of skinning.

One possible future extension may be to model spring motion using this skinning system. If we re-imagine our drag and followthrough effects in the context of spring motion, we can consider their combined result as a single oscillation. Drag models half of the oscillation, followtrough models the other half. Hypothetically, if we were to expand the system to include higher order derivatives, more number of oscillations could be modeled. We know from definitions that:

p = Displacement

$\dot{p}$ = Velocity

$\ddot{p}$ = Acceleration

$\dddot{p}$ = Jerk

$\ddddot{p}$ = Jounce (or Pinch)

These kinematic terms could be incorporated to emulate spring motion oscillations as follows:

Displacement p = result of LBS

Velocity $\dot{p}$ = first half of first oscillation

Acceleration $\ddot{p}$ = second half of first oscillation

Jerk $\dddot{p}$ = first half of second oscillation

Jounce (or pinch) $\ddddot{p}$ = second half of second oscillation

Even more higher order derivatives may be computed to yield further oscillations of the spring as desired. We see that this skinning system is not merely a collection of effects, but a framework upon which desired effects may be built. From an implementation perspective, we see from the production evaluation that artists prefer to work within Maya, if possible. Therefore, I hypothesize that implementing Acceleration Skinning as a plugin for Maya would improve artists' satisfaction as well as speed up workflow. Having Acceleration Skinning available within Maya would allow animators to immediately visualize the effects of secondary motion applied on top of their character animation. If desired, they may even switch back and forth between animating the primary motion and tuning Acceleration Skinning effects to have the two work hand in hand.

Another possible application lies in employing this technique for use in real-time gaming.

As Acceleration Skinning works out of the box, it may be applied on top of gaming character animation. This may be beneficial to automatically add secondary effects to game characters that use user input. While all of the content and images presented in this thesis have been from CPU based computation, Velocity Skinning has already been adapted for GPU accelerated computing [32]. All of the deformers presented operate on a per-vertex basis, making them simple to encode in a vertex shader.

In summary, Acceleration Skinning extends the Velocity Skinning method, tapping further into its potential, and provides a wider collection of stylized deformations for artists to employ.

Bibliography

[1] "Spooks" *Flip the Frog*, animated by Ub Iwerks, Grim Natwick, produced by Celebrity Pictures, Distributed by Metro-Goldwyn-Mayer, 1931.

[2] *Steamboat Willie*, directed by Walt Disney and Ub Iwerks, produced by Walt Disney Studio, distributed by Celebrity Productions and Cinephone, 1914.

[3] "Skeleton Dance" *Silly Symphony*, directed by Walt Disney, animated by Ub Iwerks, Les Clark, Wilfred Jackson, produced by Walt Disney Productions, Distributed by Columbia Pictures, 1929.

[4] *Looney Tunes*, produced by Leon Schlesinger Productions, distributed by Warner Bros., 1930 - 1944.

[5] Alexis Angelidis and Karan Singh. Kinodynamic skinning using volume-preserving deformations. In *SCA*, 2007.

[6] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256, 1992.

[7] Zeeshan Bhati, Asadullah Shah, Ahmad Waqas, and Hafiz Abid Mahmood Malik. Template based procedural rigging of quadrupeds with custom manipulators. In *2013 International Conference on Advanced Computer Science Applications and Technologies*, pages 259–264, 2013.

[8] Preson Blair. *Cartoon Animation.* Walter Foster Publishing, 1980.

[9] Steve Capell, Matthew Burkhart, Brian Curless, Tom Duchamp, and Zoran Popovic. Physically based rigging for deformable characters. In *SCA*, 2005.

[10] E. Catmull and A. Wallace. *Creativity, Inc.: Overcoming the Unseen Forces That Stand in the Way of True Inspiration.* Random House Publishing Group, 2014.

[11] Stephen Cavalier. *The World History of Animation.* University of California Press, 2011.

[12] Stelian Coros, Sebastian Martin, Berhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. Deformable Objects Alive! *ACM Trans. on Graphics. Proc. ACM SIGGRAPH*, 31(4), 2012.

[13] Crispin Deul and Jan Bender. Physically-based character skinning. In *VRIPhys*, 2013.

[14] Edwin Catmull, director and animator. *A Computer Animated Hand*, 1972.

[15] Marcos Garcia, John Dingliana, and Carol O'Sullivan. A Physically Based Deformation Model for Interactive Cartoon Animation. In *VRIPHYS*, 2007.

[16] James Stuart Blackton, director. *The Enchanted Drawing*, Edison Studios, 1900.

[17] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008.

[18] Ladislav Kavan and Olga Sorkine. Elasticity-Inspired Deformers for Character Articulation. *ACM Trans. on Graphics. Proc. ACM SIGGRAPH Asia*, 31(6), 2012.

[19] Jiyong Kwon and InKwon Lee. Exaggerating Character Motions Using Sub-Joint Hierarchy. *Computer Graphics Forum.*, 27(6):1677–1686, 2008.

[20] Yunmi Kwon and Kyungha Min. Motion Effects for Dynamic Rendering of Characters. *Lecture Notes in Electrical Engineering*, 181:331–338, Jan. 2012.

[21] John Lasseter. Principles of Traditional Animation Applied to 3D Computer Animation. *Computer Graphics. Proc. ACM SIGGRAPH*, 21(4), 1987.

[22] Binh Huy Le and JP Lewis. Direct Delta Mush Skinning and Variants. *ACM Trans. on Graphics. Proc. ACM SIGGRAPH*, 38(4), 2019.

[23] Gene S Lee, Andy Lin, Matt Schiller, Scott Peters, Mark McLaughlin, Frank Hanner, and Walt Disney Animation Studios. Enhanced dual quaternion skinning for production use. In *SIGGRAPH Talks*, pages 9–1, 2013.

[24] Leonard Maltin and Jerry Beck. *Of Mice and Magic: A History of American Animated Cartoons.* Plume, 1987.

[25] Joe Mancewicz, Matt L. Derksen, Hans Rijpkema, , and Cyrus A. Wilson. DeltaMush: Smoothing Deformations While Preserving Detail. In *DigiPro*, 2014.

[26] Sean Mauch. A fast algorithm for computing the closest point and distance transform. 2000.

[27] Russel Merritt and J. B. Kaufman. *Walt Disney's Silly Symphonies. A Companion to the Classic Cartoon Series.* La Cineteca del Friuli, 2006.

[28] Naoya Iwamoto and Hubert P.H. Shum and Longzhi Yang and Shigeo Morishima. Multi-layer Lattice Model for Real-Time Dynamic Character Deformation. *Computer Graphics Forum. Proc. Pacific Graphics*, 34(7), 2015.

[29] Paul Noble and Wen Tang. Automatic Expressive Deformations for Stylizing Motion. In *GRAPHITE*, 2006.

[30] Jeffrey A. Okun and Susan Zwerman. *The VES Handbook of Visual Effects. Industry Standard VFX Practices and Procedures.* Focal Press, 2010.

[31] Agata Opalach and Steve Maddock. Disney Effects Using Implicit Surfaces. In *Workshop on Animation and Simulation*, 1994.

[32] Damien Rohmer, Marco Tarini, Niranjan Kalyanasundaram, Faezeh Moshfeghifar, Marie-Paule Cani, and Victor Zordan. Velocity Skinning for Real-time Stylized Skeletal Animation. *Computer Graphics Forum*, 2021.

[33] Frank Thomas and Ollie Johnston. *Disney Animation: The Illusion of life.* Disney Editions, 1981.

[34] Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, , and Mathias Paulin. Implicit Skinning: Real-time Skin Deformation with Contact Modeling. *ACM Trans. on Graphics. Proc. ACM SIGGRAPH*, 32(4), 2013.

[35] Rodolphe Vaillant, Gaël Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. Robust iso-surface tracking for interactive character skinning. *ACM Transactions on Graphics*, 33(6):1 – 11, November 2014.

[36] Richard Williams. *The Animator's Survival Kit–Revised Edition: A Manual of Methods, Principles and Formulas for Classical, Computer, Games, Stop Motion and Internet Animators*. Faber and Faber, 2009.

[37] Winsor McCay, director and animator. *Gertie the Dinosaur*, 1914.

[38] Jiayi Eris Zhang, Seungbar Bang, David Levin, and Alec Jacobson. Complementary Dynamics. *ACM Trans. on Graphics. Proc. ACM SIGGRAPH Asia*, 2020.