UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques-Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Martin Guay

Thèse dirigée par **Marie-Paule Cani & Rémi Ronfard**

préparée au sein **du Laboratoire Jean Kuntzmann (LJK)**
et de **l'École doctorale EDMSTII**

Sketching free-form poses and movements for expressive character animation

Thèse soutenue publiquement le **2 juillet 2015**,
devant le jury composé de :

Michiel van de Panne
Professor, University of British Columbia, Rapporteur

Robert W. Sumner
Adjunct Professor, ETH Zurich & Director, Disney Research Zurich, Rapporteur

Marie-Paule Cani
Professeure, Grenoble INP, Directrice de thèse

Rémi Ronfard
Chargé de recherche, INRIA, Co-Directeur de thèse

Frank Multon
Professeur, Université Rennes 2, Examinateur

Paul Kry
Professor, McGill University, Examinateur

Joëlle Thollot
Prefesseure, Grenoble INP, Présidente

Abstract

Free-form animation allows for exaggerated and artistic styles of motions such as stretching character limbs and animating imaginary creatures such as dragons. Creating these animations requires tools flexible enough to shape characters into arbitrary poses, and control motion at any instant in time. The current approach to free-form animation is keyframing: a manual task in which animators deform characters at individual instants in time by clicking-and-dragging individual body parts one at a time. While this approach is flexible, it is challenging to create quality animations that follow high-level artistic principles—as keyframing tools only provide localized control both spatially and temporally.

When drawing poses and motions, artists rely on different sketch-based abstractions that help fulfill high-level aesthetic and artistic principles. For instance, animators will draw *lines of action* to create more readable and *expressive* poses. To coordinate movements, animators will sketch *motion abstractions* such as semi-circles and loops to coordinate bouncing and rolling motions. Unfortunately, these drawing tools are not part of the free-form animation tool set today.

The fact that we cannot use the same artistic tools for drawing when animating 3D characters has an important consequence: 3D animation tools are not involved in the creative process. Instead, animators create by first drawing on paper, and only later are 3D animation tools used to fulfill the pose or animation. The reason we do not have these artistic tools (the line of action, and motion abstractions) in the current animation tool set is because we lack a formal understanding relating the character's shape—possibly over time—to the drawn abstraction's shape.

Hence the main contribution of this thesis is a formal understanding of pose and motion abstractions (line of action and motion abstractions) together with a set of al-

gorithms that allow using these tools in a free-form setting. As a result, the techniques described in this thesis allow exaggerated poses and movements that may include squash and stretch, and can be used with various character morphologies. These pose and animation drafting tools can be extended. For instance, an animator can sketch and compose different layers of motion on top of one another, add twist around strokes, or turning the strokes into elastic ribbons. Fig. 1-5 shows a summary of the content and contributions of this thesis, also summarized as follows:

1. The line of action facilitating expressive posing by directly sketching the overall flow of the character's pose (Chapter 3).

2. The space-time curve allowing to draft full coordinated movements with a single stroke—applicable to arbitrary characters (Chapter 5).

3. A fast and robust skeletal line matching algorithm that supports squash-and-stretch (Chapter 5).

4. Elastic lines of action with dynamically constrained bones for driving the motion of a multi-limbed character with a single moving 2D line (Chapter 4).

Acknowledgments

I want to express my gratitude to my advisors Marie-Paule Cani and Remi Ronfard for giving me the opportunity to do this thesis, and for allowing me so much freedom along the way while supporting me in times of hardship.

Michael Gleicher came to Grenoble during his sabbatical, and we immediately started collaborating on character animation. I learned a great deal from his experience on the topic and am grateful for the conversations we had.

I want to thank to all the people in the lab who helped with the submissions' material and deadlines. Antoine Begault, Estelle Charleroy and Laura Paiardini helped and supported me on different projects. Antoine was able to dive into my code and implement some of the much-needed user interface, while setting-up a user-study for the space-time sketching paper. Sketching and interaction requires a lot of software design and many iterations with users. Estelle and Laura, two artists in our team, always gave me feedback about my methods—telling me whether they were useful or not, which had a positive impact on this thesis. I would ask them to gesture over animations and then I could see if my ideas for sketching motion would be natural or not to most people. Laura helped me with Maya on the *line of action* paper, and Estelle used my software to produce a large portion of the animation results in the *sketching motion* video.

I thank all the committee members Michiel van de Panne, Bob Sumner, Paul Kry, Frank Multon and Joölle Thollot for their suggestions and improvements.

It is no secret, my passion for graphics and animation is for entertainment. Before my Ph.D. and master's, I worked as computer graphics engineer doing video games. I thank everyone at Cyanide Studio in Montreal who taught me to efficiently write large scale software code. A lot of the work I did on fluids reflects my experience in videos games, providing simple to implement methods that could be easily integrated into large game engines. In practice, complexity can have a cost and simple algorithms that developers can adapt or combine with other ones are very appealing.

Working at INRIA grenoble was a great experience. There was always a nice talk or presentation to attend. In particular, I would thank Florence Bertail for organizing a reading group on solid simulation which considerably accelerated my understanding of variational calculus. During my passage at INIRA I have made fabulous friends who both

directly, and indirectly supported my research. Pierre-Edourad Landes proof-read most of my papers with a sharp eye. Mathieu Nesme with whom I had so many insightful conversations about constraints and simulation. Charles De Roussier for letting me win at battlefield. Pierre-Luc Manteaux, and Even Etem for sharing the office with me. Finally, I want to thank all the good company during conferences: Ali Dicko, Damien Rhomer, Camille Shreck, Antoine Millez and Fabian Hahn.

Contents

1 Introduction

Tools for the free-form animation of 3D characters are designed to allow exaggerated and artistic styles of movements. For example they allow using expressive devices such as squash and stretch for exaggerating movements, as well as animating iconic creatures such as dragons. Creating such animations requires *flexible* tools that allow deforming the character's shape into arbitrary poses, and control the movement at any instant in time.

The current approach to free-form animation is called keyframing. The essential idea consists in creating key poses for the character at different moments in time, and then use interpolation to produce the motion between. The name (keyframing) is inspired by traditional hand-drawn animation work flows. In hand-drawn animation, every image (frame) of the animation (i.e. film) must be drawn (e.g. 24 frames per second) [Laybourne, 1998]). What often happens however is that an expert animator will draw only important *key* frames, and then armies of animators will draw the frames between.

In 3D animation, the character is not created from scratch at each frame, but instead re-used via deformation. 3D characters are typically represented with a surface mesh which can take a long time to create from scratch (a process called character modeling). Once the character is modeled it is animated like a puppet. With keyframing tools, the animator sets the character into poses at different key times, and then interpolation is used to produce the motion between the different key poses.

Posing a 3D character by deforming every vertex of the surface mesh would be extremely laborious. Instead, several tools were devised to provide control over areas of the character's surface. For example, a skeleton or cage (see different *rigs* in Fig. 1-1) maps the vertices of the surface and the animator manipulates only the reduced bones or cage nodes by clicking-and-dragging to deform the character. While there has been

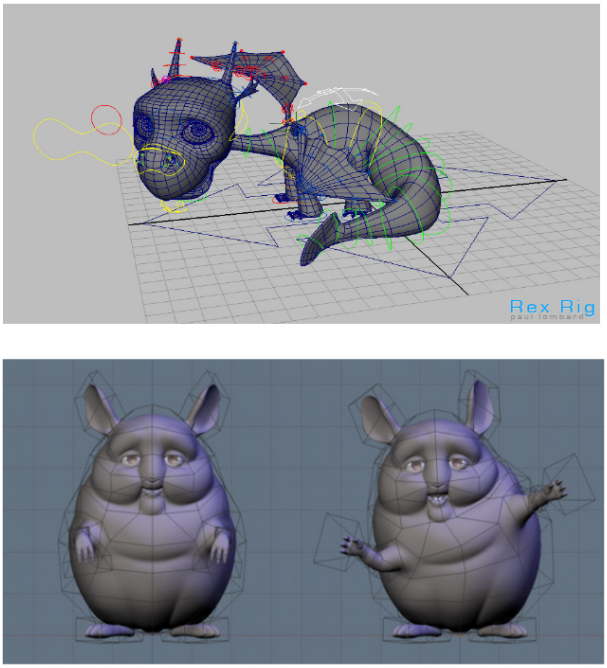

Figure 1-1: Top is a traditional rig used in keyframing 3D animation. Different FK/IK controls are manipulated to the deform a character's shape. Bottom image is a free-form deformation (FFD) cage-based rig. In both cases, the user clicks and drags on the nodes (e.g. bone or lattice) to deform the character.

a fair amount of research on different rigs and deformers, the main controller used in production today is the skeletal rig.

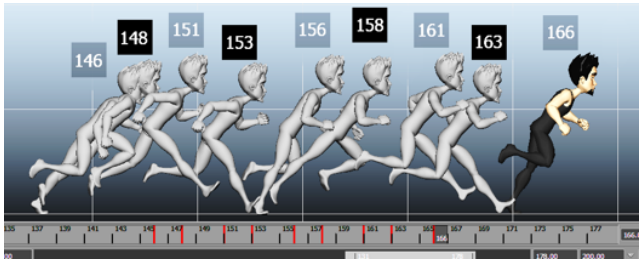

Figure 1-2: With keyframing, the animator creates different poses at key times, and the motion between the keyframes is produced by interpolating the key poses. Coordinating a motion involves a back-and-forth process between selecting times and editing the pose spatially.

Deforming the shape of the character into a pose is one part of the keyframing process. The other consists in setting the timing of the *keyframe* via the timeline (see bottom of Fig. 1-2 for a timeline). The motion is produced by interpolating the degrees of freedom controlling the character (e.g. the angles of the skeleton's joints). Typically a

back-and-forth process between editing the shape and editing its timing occurs when coordinating the character's motion.

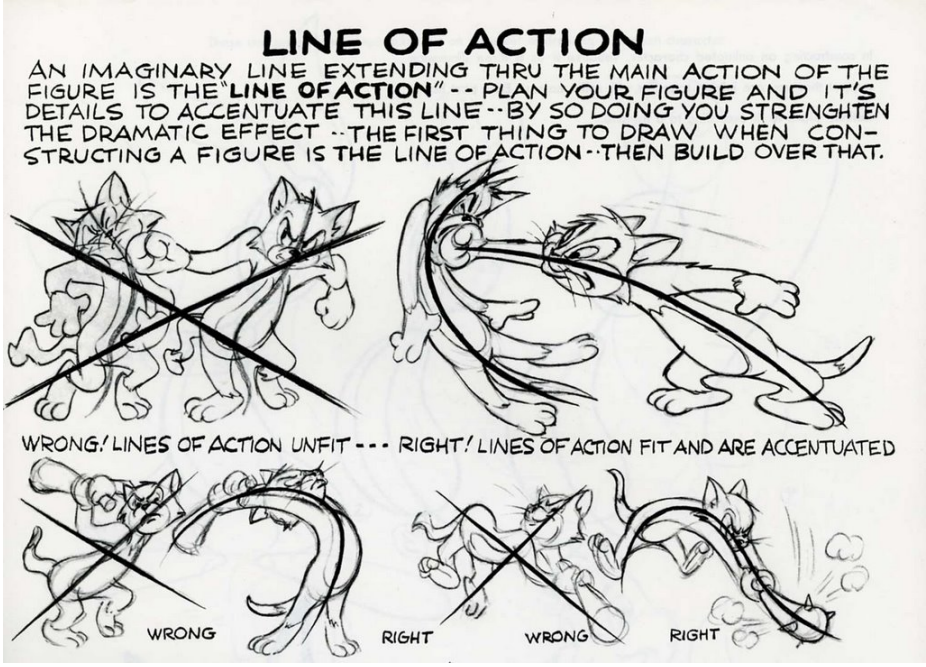

Figure 1-3: The line of action is a tool that helps drawing more expressive and readable poses. It helps provide the pose with a coherent and aesthetic overall flow, making it easy for the viewer to understand the action, intention or state of mind expressed by the pose. The tool consists in first drawing the line of action, and then populating the body of the character around the line of action—as to ensure a consistent overall flow. In this thesis, I provide a mathematical definition of the line of action which can be used to pose 3D virtual characters by sketching a line of action. Image from Preston Blair's book Cartoon Animation [Blair, 1994], ©The Estate of Preston Blair.

While the current keyframing tools are very flexible, using them to create quality animations is time consuming and laborious. The more keyframes used to represent the motion, the more coordination is required by the animator. The number of keyframes for a motion is often the metric used to measure the quality of animations—given the same talented animator. For example, TV quality animation uses less keyframes and the productivity averages between 25-30 seconds of finished animation per week (per animator). Animated feature films tend to have almost every frame as a keyframe, and the productivity is as low as 3-4 seconds of animation per week [Riki, 2013]. For games, animation productivity ranges between 5-10 seconds per day, or 25-50 seconds per week. Loosely, the quality of the animation has to do with the level of detail, both in the number of keyframes used to represent the motion, and the dimensionality of the character's

shape (e.g. the number of "bones", and the quality of deformations).

One of the main reasons it takes so long to keyframe animations is that current posing and coordination tools—while designed to be very flexible and allow any motion to be produced—were not designed to provide direct control over high-level aesthetic and artistic principles. For instance, when manipulating a skeletal rig, we can very easily violate the artistic principle of flow, as shown by the cartoon character in Fig. 1-3. The flow of a pose is the reading through the pose. A clear flow helps viewers understand the pose more easily and rapidly. If an animator wants to create an aesthetically pleasing flow for a posture, he must manipulate a set of widgets along the skeleton so as to fulfill the aesthetically pleasing curve. In other words, providing the pose with an overall coherent flow is only indirectly supported by current posing tools.

Additionally, quality animations are not just a set of aesthetic keyposes, but involve the careful coordination of shapes over time. However, current keyframing tools are focused on a single instant in time (the actual keyframe), and do not provide control over the coordination as a whole. To coordinate the movement the animator must edit different keyframes separately. Moreover, the very way the keyframes are interpolated renders certain styles of movements less accessible. In particular, coordinating shapes into *following paths* requires creating nearly every frame of the animation as a keyframe when using "point-to-point" interpolation. For instance a stylized dance may include waving the arms. In this case, the animator would have to create nearly every frame of the animation as a keyframe. This is a problem because the more keyframes there are, the more challenging it becomes to coordinate the motion. In consequence, animators tend to create less of these styles of movement.

These two problems, indirect control over the flow of a pose and indirect control over coordination, exemplify how current free-form animation tools lack direct control over high-level aesthetic principles. As a result, animators chose to create by first drawing on paper instead of creating directly with 3D animation tools. If we look closely at how animators draw expressive poses and coordinated movements, we can see tools that help them fulfil high-level aesthetic and artistic principles.

For instance, to provide a pose with a more coherent and easily readable flow when drawing, artists use the line of action. They first draw a stroke directly specifying the flow of the character, and then populate the body (the details) around it. The line of action is

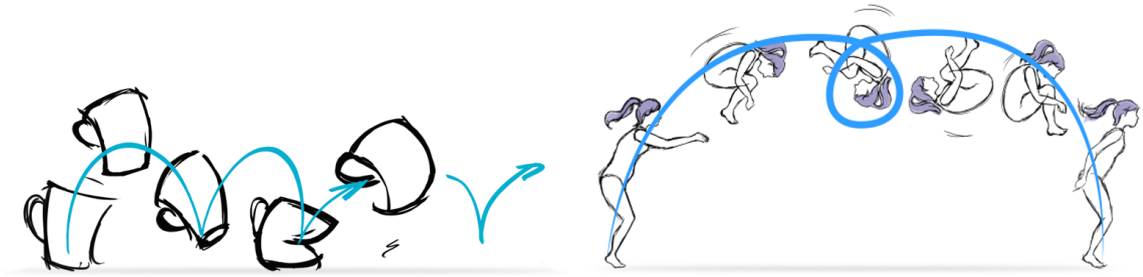

Figure 1-4: Animators often establish coordination by drawing motion abstractions on paper (hand-drawn). Unfortunately, these abstractions cannot be used with current free-form animation tools (i.e. keyframing). Hence animators create first on paper, and then only later fulfill the motion with computer animation tools (with keyframing). The reason is because we lack an understanding of these motion abstractions relating the character's shape over time to the drawn abstraction. In this thesis, I introduce a formal understanding of these motion abstractions that allow using them directly to animate arbitrary characters.

directly linked to the way the viewer will read the pose, i.e. the way the viewer's eye will flow through the pose. Unfortunately, the line of action is not part of the current free-form animation toolset. To fulfill a line of action with a 3D character, animators must coordinate different parts of the skeleton into forming an aesthetic curve shape—instead of directly drawing a line of action.

To achieve quality coordination, animators will often sketch on paper motion abstractions (e.g. bouncing or rolling shown in Fig. 1-4). These help get a better sense of the timing and spatiality of the motion. Again realizing the coordination with keyframing is indirect: after drawing on paper, the animator uses the posing tools and the timeline to fulfill the abstraction with the character's motion—instead of directly sketching an abstraction to set the character into motion.

Without directly considering artistic goals, computer animation tools may never be involved in the creative process. Hence it is important that we better understand artistic and aesthetic goals in order to devise tools that allow people to more directly create quality animations.

Challenges

The goal of this thesis is to ease the creation of expressive poses and coordinated movements. My approach consists in modeling commonly drawn abstractions of shapes and movements (line of action, paths, semi-circles and loops), resulting in sketch-based tools

that directly operate at the artistic and aesthetic level. With these tools, animators are able to directly sketch the overall flow of 3D character with a single stroke, as well as specify a full coordinated motion by sketching a single motion abstraction. The first challenge in modeling these abstractions and turning them into computational tools lies in our ability to describe them mathematically.

There are many books on lines of action, but never was the concept formally defined. In other words, it is hard to describe the character's shape as a function of the drawn stroke. Similarly for motion abstractions, we all understand what they mean (bouncing, rolling), but it remains a challenge to write the character's shape over time as a function of the path's shape. Understanding these mathematically not only advances the body of knowledge on animation, but allows the derived computational tools to be applied to arbitrary character morphologies.

The other major challenge is the lifting of 2D projective curves into 3D curves (used to automatically deform the character's geometry via its skeleton parametrization). This problem is ambiguous and under-constrained. The real challenge for free-form animation is to lift the 2D sketch without sacrificing the expressiveness of the devised tool. As in traditional hand-drawn animation, the stroke should be able to stretch the character, and describe unrealistic poses and motions. Additionally, motion abstractions lead to dynamic 2D curves, which introduces a new issue: temporal continuity must be ensured in a robust manner.

Contributions

The research presented in this thesis eases the creation of expressive poses and facilitates the creation of coordinated movements via gestured strokes. The resulting techniques are not bound to specific body parts or morphologies, allow exaggerated poses and movements and can therefore be used in the general setting of free-form animation. My approach is focused on sketching shape and motion abstractions in the form of curves: the static line of action (LOA)—specifying a pose—and the abstract *space-time curve*, that is turned into a dynamic line of action (DLOA) to drive the character's body over time.

In all cases, a line matching procedure is required to lift the 2D curve (dynamic or static) into 3D; in order to ultimately deform the character's geometry via its skeletal

parametrization. Over the course of the thesis I have proposed various solutions to this problem, culminating to a fast and robust method that combines closed-form solutions with dynamic programming to allow squash and stretch, as well as unrealistic bendings and motions.

The line of action is a gestured stroke that deforms a set of bones in the character's body as to match the drawn line. The first contribution described in Chapter 3 is the mathematical definition of the line of action, general enough to encompass any body part; including non-humanoid parts such as the tail of a dragon. I discuss how to establish a correspondence between the bones and the drawn stroke.

The space-time curve allows specifying a full coordinated motion with a single stroke. This approach is based on the insight that commonly drawn motion abstractions such as bounces and rolls (Fig. 1-4) encode information of shape over time. This allows establishing a direct relationship between the character's shape over time and the shape of the drawn stroke (described in Chapter 5). The character's motion is abstracted through a dynamic line of action, used as a projective constraint driving the character's motion. The same DLOA projective constraint can be applied to any body part or character. Finally, DLOAs can be composed together leading to control over complex motions with few strokes.

Matching the shape of 3D characters to the shape of 2D curves is ambiguous and under-constrained—many 3D poses can match the 2D line. To remove depth ambiguities, I supplement the problem with a viewing plane constraint (introduced in Chapter 3). This allows the user to create virtually any pose in the viewing plane, but requires the user to rotate, in order to deform the character in depth. In Chapter 3, I present a matching algorithm that solves for the orientations of the bones as to match the LOA using non-linear programming. This first solution has several limitations: it does not support squash and stretch and when applied sequentially to a moving line, leads to inconsistencies between frames (noted in Chapter 4). In Chapter 5, I introduce a robust matching method based on closed form solutions combined with dynamic programming. This method allows squash-and-stretch and solves the problem exactly, thereby guaranteeing consistency between frames of the animation.

Twist cans. Because the LOA only conveys 2D information of shape, artists often establish the orientation around the line by sketching orthogonal strokes, or in more explicit cases, a pair of can-shaped primitives. Twist cans are introduced in Chapter 4.

Physically-based LOA. I explored the idea of using a physically-simulated line of action to drive the motion of the character. With a freely moving LOA driving the motion of an articulated character (e.g. biped), we need to consider changes in body parts driven by the line over time. In Chapter 4, I take a first step towards automating this for animators with a method that propagates the sparsely annotated body attachments to the rest of the motion.

Publications. This work, produced over a period of two years and a half has lead to the following publications:

- *The line of action: an intuitive interface for expressive character posing*, Martin Guay, Marie-Paule Cani and Rémi Ronfard, ACM TOG, *In proceedings of SIGGRAPH Asia 2013*.
- *Adding dynamics to sketch-based character animations*, Martin Guay, Rémi Ronfard, Michael Gleicher and Marie-Paule Cani. *In proceedings of Expressive 2015, and the 5th Symposium on Sketch-Based Interfaces and Modeling (SBIM) 2015*.
- *Space-time sketching of character animation*, Martin Guay, Rémi Ronfard, Michael Gleicher and Marie-Paule Cani, ACM TOG, *In proceedings of SIGGRAPH 2015*.

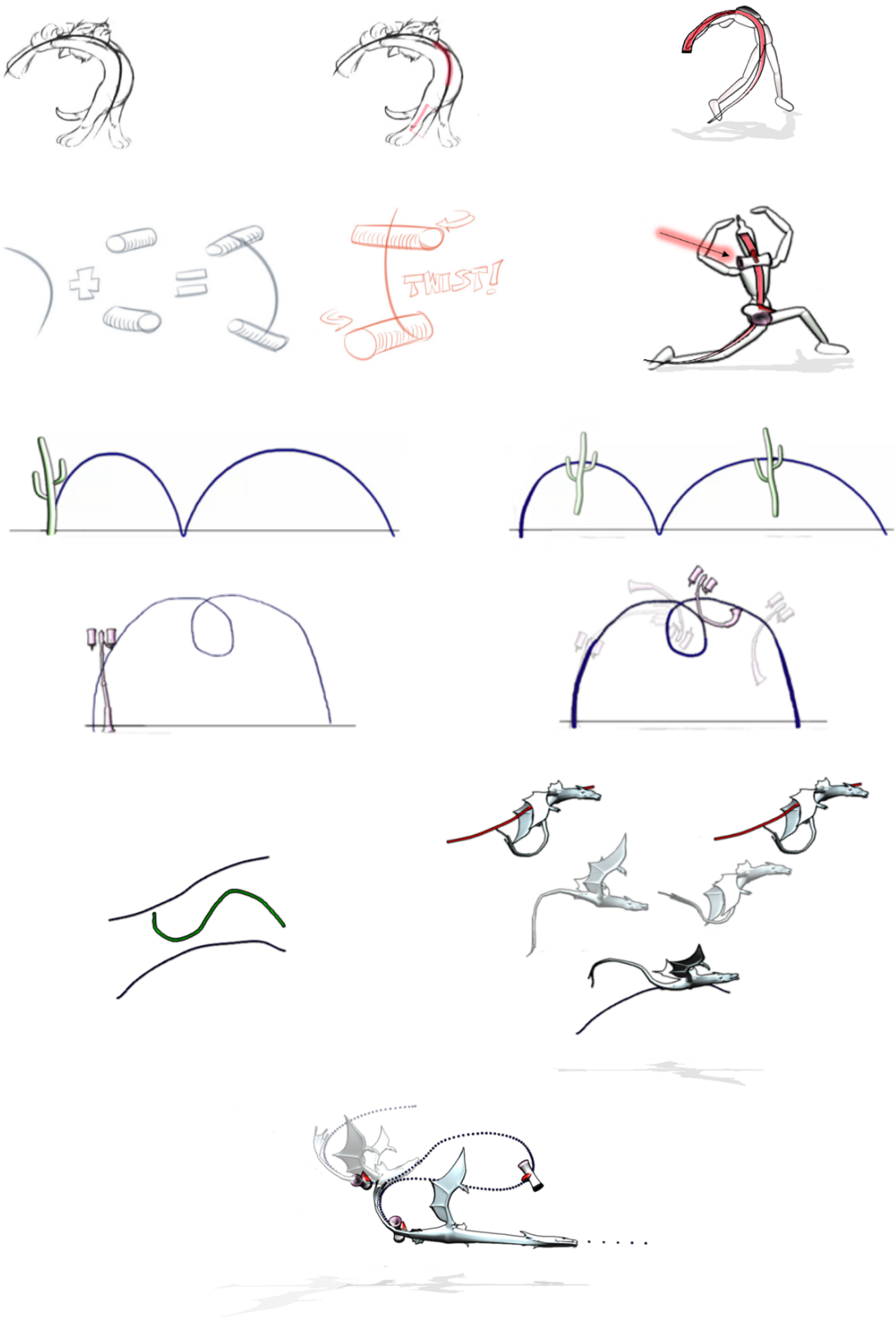

Figure 1-5: Overview of this thesis: first row is the line of action definition, second row is the "two cans" drawing technique to specify twist, third and fourth row are the motion abstractions for arbitrary characters, fifth is motion composition with path-following styles of movements and sixth row shows a combination of stretchy paths (neck of dragon elongated) with twist cans.

2 | Background

This chapter provides a large overview of computer animation starting with the artistic principles, the basic representations of 3D digital characters and the tools to animate them. The general goal of this thesis is to make the creation of quality character animation more intuitive and natural. While many previous works cope with this problem by using pre-existing motion, I choose to focus in this thesis on the tools and techniques for free-form animation—in particular of body animation (as opposed to facial animation for example). Methods for free-form animation should allow exaggerated, unfeasible and artistic movements, including imaginary non-humanoid characters.

Hence I start by covering the set of tools that meet this level of flexibility and is most relevant to this thesis. These include keyframing and its dual: partial performance (or acting). Keyframing freezes time and allows the animator to change the character spatially—using different modalities (clicking-and-dragging, sketching, or manipulating tangible devices)—while partial acting freezes space (choose 1 degree of freedom) and the animator performs this DOF over time.

Because these methods are laborious and time-consuming, many researchers have used data (mostly human motion) to make the creation of animations more intuitive. For instance, by using a model of human poses—learned from a database—one can fill the missing parts of a 3D pose that are only partially specified (e.g. with a 2D sketch). Unfortunately, these strong assumptions about the character's space of poses and movements limits the possibilities the tool will permit—not to mention the fact that most animation databases are comprised of human movements. While limited, it can be useful for quick prototyping and drafting of motions. Applications could consider these restrained motions for increased intuitiveness with the possibility to use motion editing tools such as [Gleicher, 1997, Lee and Shin, 1999] for refinement.

Another appealing idea to fill the missing motion is to find the most physically plausible motion in a so-called space-time constraints framework. Typically, these formulations model the mechanics of the character, and minimize an energy consumption objective. What is appealing with this formulation is that it could assist the animator in creating more realistic and dynamic motion, while allowing to bend the physics if necessary. Unfortunately, in practice these models include very limited information about the character's intentions, and trajectory optimization quickly becomes high dimensional and subject to the curse of dimensionality: local optimization often results in a local minimum.

1 Principles of animation and art

Disney's 12 animation principles, popularized in computer graphics research by John Lasseter in 1987 [Lasseter, 1987], are well documented on the internet and taught in every entry level class in computer animation. While they have been known to researchers and engineers for a long time, very few of these principles are directly taken into account by the free-form animation toolset.

Arcs The principle of arcs requires objects to follow arced trajectories rather than straight lines is automatic when interpolating the angles of articulated characters. To some extent, it is actually harder to create straight line trajectories with 3D articulated characters, than it is to create arc trajectories.

Anticipation, follow-through and overlap One way animators include anticipation, follow-through and overlap (or successions) in their animation is by copy pasting keyframes and setting an offset in time. This process can be automated by warping within the parametric space of the keyframe splines [Neff and Fiume, 2003].

Timing In traditional animation, timing is the number of frames or drawings in the film. When keyframing digital characters, timing is related to the number of keyframes and their respective time. It is the timing of the keyframes in relation to the spatial distance of the poses or shapes that provides the velocity (and acceleration) of the motion. Basic animation effects such as ease-in and ease-out can be created by manipulating the

full body timing alone. However, quality animation involves overlaps and successions with requires coordinating both the relative timing between body parts as well as their spatiality. The interplay between timing and shape is the art of coordination and to some extent, the heart of animation. In fact entire books [Williams, 2002] have been dedicated to this interplay.

Exaggeration helps the viewer understand the intentions and actions of the character. For example, the animator can exaggerate the arcing of a character that is throwing a ball to create anticipation, or he can squash or stretch the character's limbs.

Squash and stretch Considered by many as one of the most important devices in animation, squash and stretch is used to give the illusion of weight and volume to a moving character, as well as build anticipation and follow-through. Creating squash and stretch requires volume preservation which is addressed by many works including [Rohmer et al., 2009] with local skinning. The tools I present in this thesis allow stretching the bones of a character, but the skin deformation technique used to produce the results (linear blend skinning) does not preserve volume.

Staging and appeal The actions of the characters and their placement in the scene should be staged as to ease the readability of the animation as well as entice the viewer to keep watching. The characters should be posed in aesthetically pleasing manners that facilitate the viewer's understanding of the characters' intentions and actions.

While animation principles are often cited in computer animation research, the more general principles of art [Pri, 2015] tend to be overlooked. One of these principles is called movement (or flow) and is very much relevant to this thesis.

Movement (or flow) is the path the viewer's eyes follow when looking at a piece of art. *"The artists control what the viewers see and how they see it, like a path leading across the page to the item the artist wants the viewer's attention focused on"* [Pri, 2015]. In this thesis I will use the term flow instead of movement, which is also a term used in many guidebooks on drawing. An aesthetically pleasing and coherent flow of a pose—and perhaps motion—will make the poses and animations more readable and appealing to the viewer.

2 Free-form posing

Inspired from the hand-drawn animation tradition, computer animation of characters are represented as a sequence of poses in time. To this day, specifying each pose individually and controlling the timing through a timeline remains the established work flow and pipeline. As character shapes can be highly detailed, e.g. hold thousands of vertices, a great deal of research and engineering was invested into simplifying the representation of characters, in order to facilitate their manipulation. Perhaps because mouses and keyboards were the most widely spread input devices on desktops, methods focused on the "click-and-drag" metaphor to deform the character, focusing on the idea of reducing the number of nodes to edit, in order to create a pose at a keyframe.

2.1 Clicking-and-dragging with rigs

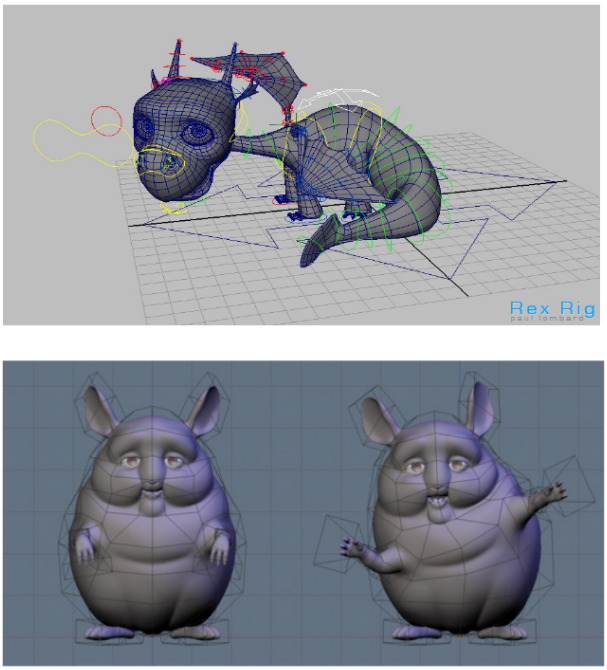

Figure 2-1: Top is a traditional rig used in keyframing 3D animation. Different FK/IK controls are manipulated to the deform a character's shape. Bottom image is a free-form deformation (FFD) cage-based rig. In both cases, the user clicks and drags on the nodes (e.g. bone or lattice) to deform the character.

One way of reducing the number of degrees of freedom to manipulate is reduce the number of vertices on the surface mesh using a subdivision scheme, along a displace-

ment map to represent surface details (as in z-brush). These technics are widely employed in production today, both for animated movies, and real-time games. However, they are not used for manipulation, but only for computation, as they still retain too many vertices to be effective. Hence, further reduction of the shape is achieved through *natural* subspaces, such as linked skeletons [Burtnyk and Wein, 1976], and lattice cages [Sederberg and Parry, 1986], parameterizing the surface geometry with fewer degrees of freedom to manipulate. Typical subspace rigs are shown in Fig. 1-1.

Although cages have been used in some productions, and are still being used for specific tasks, the skeleton remains the most widely employed subspace for deforming characters. However, manipulating every bone of a skeleton to obtain a final shape can be time consuming and non-intuitive. The relationship between the shape, and position of the skeleton is non-linear w.r.t. to the angles of the bones. Obtaining a given shape—such as being aligned onto an aesthetic curve—or obtaining a given end-effector position, involves a back-and-forth adjustment process, where the user manipulates different angles along the kinematic chains of the skeleton. To circumvent this problem, inverse kinematics (IK) methods based on optimizing high level objectives such as end-effector positions and orientations [Zhao and Badler, 1994] allow the user to manipulate fewer degrees of freedom, while the system solves for the underlying degrees freedom (angles), automatically.

These specialized control rigs remain the established way of posing today. However, they require setting up a new specialized rig each time a new character is to be deformed. Many researchers have addressed this problem [Sumner and Popović, 2004, Igarashi et al., 2005, Sorkine et al., 2004]. Deformation transfer [Sumner and Popović, 2004] allows transferring the pose of one character, onto another similar character by minimizing their local affine transformation difference over all the vertices of the mesh. Instead of transferring transformations onto another mesh, as-rigid-as-possible shape deformation [Igarashi et al., 2005, Sorkine et al., 2004] minimizes the distance to the nearest local rigid transformation of the elements of a reference shape, similarly to an elastic energy function for elastic materials.

Since the early works of [Burtnyk and Wein, 1976] and [Sederberg and Parry, 1986] on skeletons and cages, we have seen considerable efforts towards improving the quality of the deformation (e.g. [Jacobson et al., 2012] and [Lipman et al., 2008]), as well as the

now the possibility of removing specific rig setups [Sumner and Popović, 2004, Igarashi et al., 2005, Sorkine et al., 2004]. In comparison, fewer works have looked at the problem of making the specification of the pose more natural and intuitive. In the next section, we will look at the ways people can sketch stickfigures to create poses.

2.2 Sketching stickfigures

As we enter the realm of sketching, it can be interesting to see the success sketch-based modeling of 3D shapes has had over the past decade. Sketching facilitates the creation of 3D shapes by allowing users to sketch the visible contours of shapes in 2D, in order to create a 3D object or character [Igarashi et al., 1999, Karpenko and Hughes, 2006]. To create an animation, the user could always choose to create a new character at each frame, but this would be time consuming and the resulting shape would not be continuous across frames (which could always be a stylistic decision). Instead, as we have seen, the character is rigged with a skeleton, or cage, and it is the configuration space of the rig at the key poses that is interpolated.

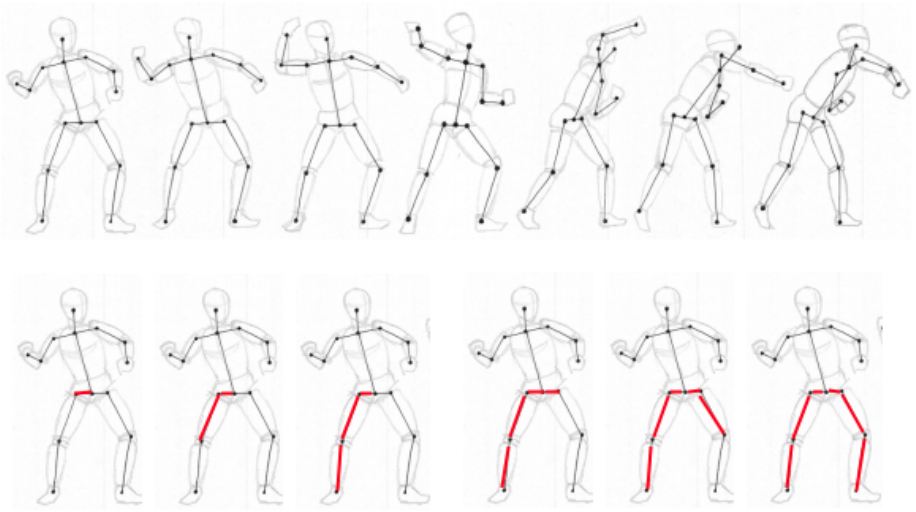

Figure 2-2: Davis (2003), asks the user to sketch the limbs one by one. An alternative in their system is for the user to draw on paper dark points for the joints. The paper is then scanned and the 3D joints—as well as skeletal structure—recovered.

To rapidly depict movements, people often use stickfigures. Luckily, this also reflects the traditional skeleton used to parameterize the surface of virtual characters. Hence for sketching character poses, the research community has investigated sketching stickfig-

ures [Davis et al., 2003, Mao et al., 2005, Lin et al., 2010, Wei and Chai, 2011, Choi et al., 2012] as an abstraction of the character's shape. Parsing the structure, or segmenting a 2D stickfigure that has been drawn in a free-form fashion is a challenge in it-self. Both [Davis et al., 2003, Lin et al., 2010] do not sketch the stickfigure, but instead model it by clicking and dragging, as shown in Fig. 2-2. The result is a set of piecewise-rigid sticks. [Mao et al., 2005] lets the user sketch limbs which are snapped to the parent limb, but this requires the user to sketch the figure in a pre-defined order. [Wei and Chai, 2011] sketches pre-defined body parts such as the spine, legs, and arms in a free-form manner, but the user is required to specify which part is being sketched. The pose can be refined by placing additional 2D projective point constraints at end effectors, for the head or foot. [Choi et al., 2012] sketches free-form body parts of the character in a pre-defined manner, but the sketch is used to recover pre-existing motions in a data-base.

While some of these methods take curved strokes as input, every method described so-far solves for rigid bones. Curved shapes, as well as squash-and-stretch—an important animation principle—have never been considered with a stickfigure interface. [Kho and Garland, 2005] sketches deformations of individual 3D limbs by first drawing a reference stroke over the 3D mesh, and then by re-drawing a second stroke, specifying the deformation. They use a flexible deformation model based on ARAP [Igarashi et al., 2005, Sorkine et al., 2004], which allows curving and stretching the mesh in a free-form manner. However, their work does not extend to full body curves such as lines of action. The work of [Kho and Garland, 2005] is much more focused on posing rather than on animating—relying solely on the mesh does not provide a natural way to interpolate the poses; in contrast to skeletons where the configuration space can be efficiently interpolated, allowing animations to be created with few poses. The need to recover a 3D skeleton from the 2D stickfigure, which is a highly ambiguous problem—has been the main focus of the literature since [Davis et al., 2003].

Lifting 2D sketches into 3D

Given the semantics of the 2D strokes (the body part they correspond to), the final stage consists of lifting the 2D strokes into a 3D skeletal pose. For piecewise rigid skeletons, the problem is formulated as finding the angles of each joint in order for the 3D pose to match the 2D stickfigure in screen space. Now, re-constructing a 3D pose from a 2D

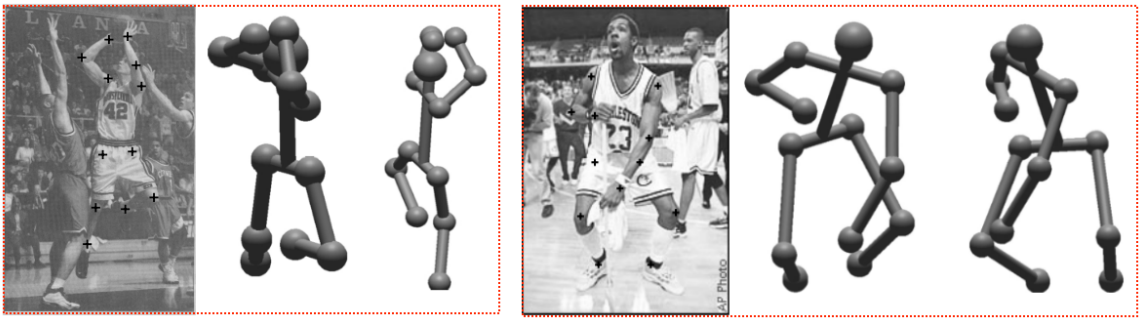

Figure 2-3: Two lifting (3D re-construction) examples from Taylor et al. 2000. The method recovers joint positions by inverse projecting the 2D points into 3D assuming an orthographic projection. The multiple possible poses are pruned based on the length of the bone segments. The second image is the same pose in a different view.

sketch is, due to depth, under-constrained; many poses exist for the same 2D stickfigure.

Works in computer vision tackled the problem of recovering 3D joint positions of a human skeleton from a marked 2D image. [Lee and Chen, 1985] and [Taylor, 2000] recover 3D positions by placing an initial pose in the scene and inverse projecting the 2D joint positions to recover their depth position (shown in Fig. 2-3). Each segment is solved one by one going down the tree structure of the skeleton. Multiple solutions satisfy the inverse projection, and a pruning process is then applied to find the most likely bone segments. Typically, two depth values are possible (the bone segment is towards the camera or away from the camera). These are pruned by comparing the resulting length of the bone segment with the initial bone length, as well as by comparing joint angle limits with those of mechanically feasible human joint angles.

In the setting of stickfigure sketching, Davis et al. [Davis et al., 2003], uses the method of [Taylor, 2000] to recover the depth value of each joint position assuming an orthographic projection, and the insights of [Lee and Chen, 1985] to prune the solutions. By using assumptions of joint angle limits for human characters, as well as measurements of balance, they significantly reduce the possible poses. The remaining poses are then presented to the user, as to let him select the most satisfying pose. To create smooth motion over the set of poses, they optimize for smoothness, while seeking to respect the image space constraints.

It is worth noting that for animation, it may be useful to let the length of the limbs stretch, as squash and stretch is an important and desirable feature in animation. Davis [Davis et al., 2003] mentions they allow longer limbs, but the results do not demonstrate

extreme squash and stretch. Also, animators tend to sketch few keyframes, which are then expected to be interpolated. Interpolating the angles of the skeletons, instead of the joint positions will preserve the lengths of the bone segments under large deformations (e.g. with few keyframes). With these methods that only recover 3D joint positions at the keyframes, it would require recovering 3D joint angles which could be done with a skeleton registration method.

In conclusion, the inverse projection methods [Lee and Chen, 1985, Taylor, 2000, Davis et al., 2003] require joint limits and user intervention to select the best pose.

In the vision community, the related problem of matching 2D pictorial elements connected by springs assembled into the structure of a humanoid are matched to pictures of human poses in [Felzenszwalb and Huttenlocher, 2005]. They find the global solution with a dynamic programming approach, including the adjustment of the size of the pictorial elements to the size of the person's limbs in the image. While this solution shows it is possible to robustly solve for 2D matching problems, their solution has not been extended to match 3D characters.

2.3 Stop motion armatures and puppets

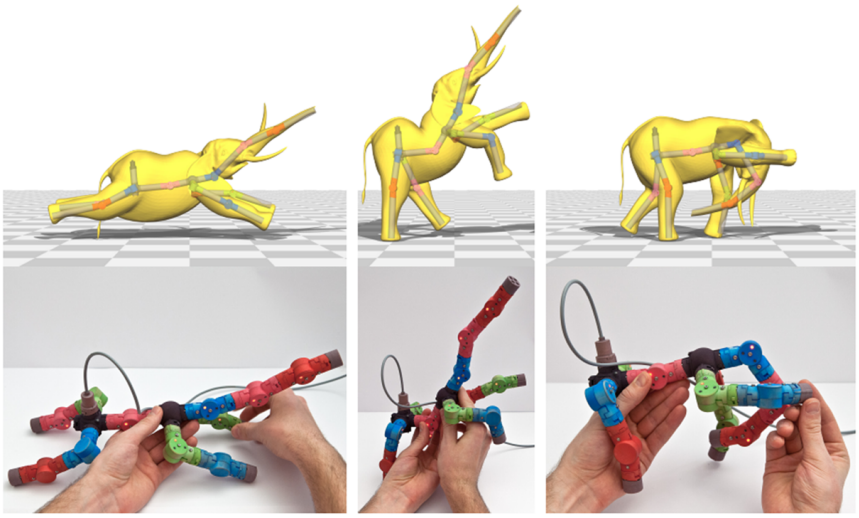

Figure 2-4: While there are many puppets to control 3D virtual characters, they are often confined to a specific morphology. Recently, a set of modular—lego-like—blocks can be assembled according to arbitrary morphologies such as an elephant.

Stop motion armatures [Knep et al., 1995, Esposito et al., 1995, Yoshizaki et al., 2011, Jacobson et al., 2014] remove the depth ambiguity completely by utilizing a 3D tangible

puppet which the user manipulates with his hands. They are used to record individual pose keyframes by deforming the puppet device into poses and taking snapshot. In the early days of 3D character animation, navigating in 3D was new to most artists. Hence for the movie Jurassic Park, they devised a T-Rex stop motion puppet [Knep et al., 1995]. With this type of tool, each new character morphology requires a new puppet to be designed, such as a Monkey [Esposito et al., 1995]. An actuated device can be used to load pre-existing virtual poses, as well as to add stiffness in the joints providing force feedback [Yoshizaki et al., 2011]. Today humanoid puppets such as the Qumarion are commercially available. Recently, [Jacobson et al., 2014] have combined the idea of using building blocks of modular limbs to assemble arbitrary morphologies with the possibility of 3D printing the modular pieces. One of the limitations of these methods is that it is not possible to stretch the limbs.

3 Free-form performance: partial acting and layering

We have seen the various ways of creating motions by keyframing poses. Each pose is selected via a timeline, and deformed individually; either with "click-and-drag" rigs, or with tangible devices. Yet, all of these tools operate at the individual pose level, and do not help coordinate movements, or provide control over the temporal domain. Providing control over space and time in a free-form fashion with low-dimensional input devices available on a desktop is a challenging problem. One possibility is to separate the character's motion into different layers and control each layer individually over time [Oore et al., 2002, Dontcheva et al., 2003, Neff et al., 2007, Shiratori et al., 2013].

Instead of sequencing poses, researchers have proposed to use hand-held motion capture devices, and to act the motion of selected degrees of freedom separately [Oore et al., 2002, Dontcheva et al., 2003, Neff et al., 2007, Shiratori et al., 2013]. The partial motion recordings are layered in order to create the full body motion. This allows to animate arbitrary characters, and to control 3D orientations with the hands.

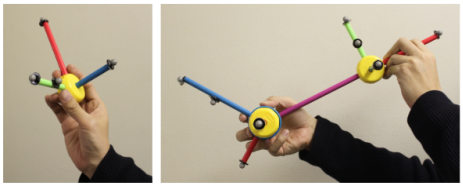

Figure 2-5: In Shiratori et al. 2013, the user selects degrees of freedom in the character, and performs he trajectory using a 6 DOF (position and orientation) TinkerToy setup held in his hand.

Before detailing each method, we can already discuss the fact that such devices are held up in the air, which can be tiring in the long run. While it is more natural to control trajectories, it becomes hard to synchronize the trajectories into meeting individual poses.

In [Oore et al., 2002], two 6D motion capture devices are presented to the user. The user selects body parts such as the two legs of a humanoid, and the mappings between the 6 degrees of freedom of the device are mapped to the degrees of freedom of the virtual character. The paper presents pre-defined mappings for humanoid characters only. Instead of using specialized 6D capturing devices, Neff [Neff et al., 2007] uses the widely available mouse and keyboard. The user creates 2D maps controlling different DOFs in the body, and then performs their motion by scrolling the mouse, and can switch between maps by pressing keys. Similarly, [Shiratori et al., 2013] use TinkerToys 6D devices (see Fig.) to control separately selected degrees of freedom, which can also be synergetic, e.g. control both elbows with the same widget.

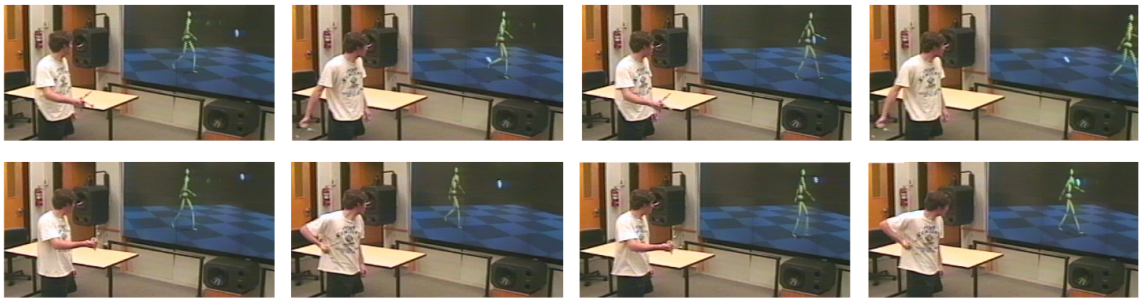

Figure 2-6: In Dontcheva et al. 2003, the user acts on top of an existing arm swing motion to establish a mapping (first row). Then he re-performs the motion a second time to edit the movement, with bent elbow.

[Dontcheva et al., 2003] adds an automatic way of selecting body parts by performing on top of an existing motion. Once the mapping detected, the user performs a different motion which controls more degrees of freedom. They apply canonical correlation analysis to automatically establish a mapping between a single 3D widget (position) trajectory, and a higher number of DOFs. They demonstrate this capacity with a spider motion where CCA detects a mapping to six legs, and the user can edit the six legs at once. The Fig. 2-6, shows this for an arm swing where the bending the elbow is inferred based on the new widget trajectory. This method could also be used with other devices such as a mouse or stylus.

4 Example-based animation

By modeling the space of human poses or motions based on a database of motions, it is possible to significantly reduce ambiguities in 3D reconstruction problems for sketching poses [Grochow et al., 2004, Wei and Chai, 2011] and motions [Min et al., 2009]. Also, with the advent of consumer depth imagery (e.g. the kinect [Shotton et al., 2011]), the idea of re-targeting human skeletal motion to arbitrary characters has had an increase in attention [Yamane et al., 2010, Chen et al., 2012, Seol et al., 2013, Rhodin et al., 2014]. The human skeleton can now be used as an interface for animating non-humanoid characters.

4.1 Sketching poses

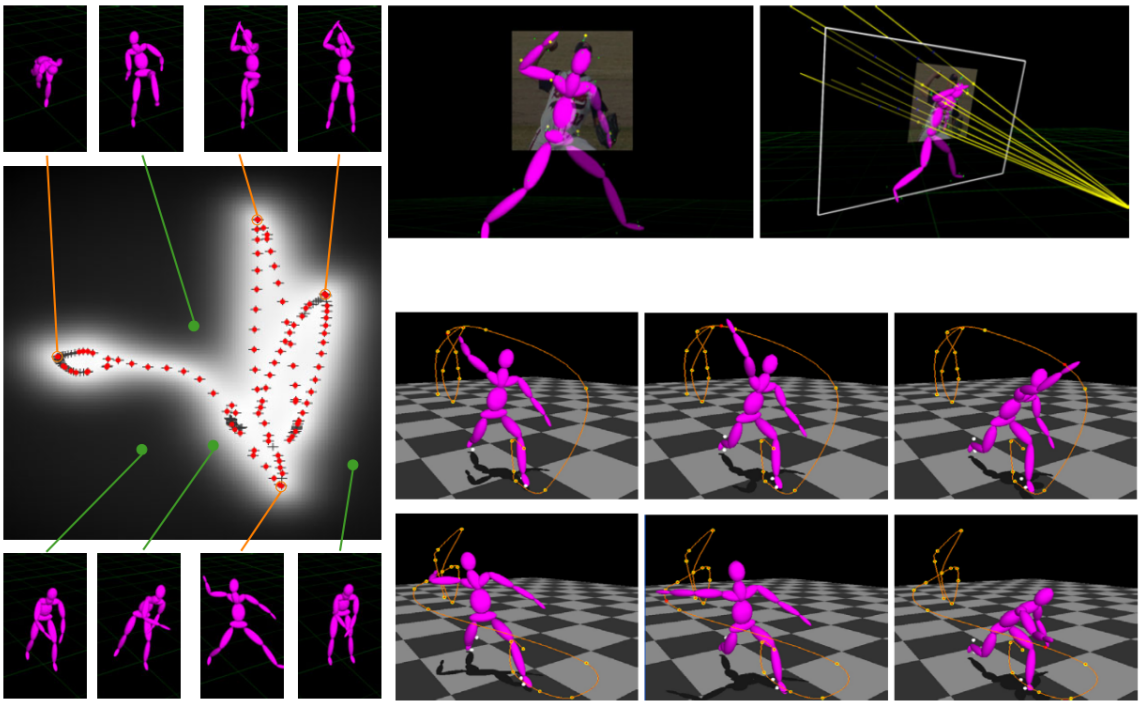

Figure 2-7: An example of the likelihood function (pose prior) for the 2D low-dimensional embedding from the SGPLVM from Grochow et al. 2004. The black areas correspond to low confidence poses and the mean pose when converted back into the character configuration space. On the right we see applications where the prior function is used to steer poses towards a likely pose in the active set (in this case a throwing motion). Top right is a 2D-3D point matching scenario and bottom right is an inverse kinematics problem. Note the poses may be different from the motion in the data-base, since the prior is only softly steering the poses towards statistically favorable poses.

To constrain the space of possible poses, we can learn a model of poses from existing data and use it as a prior in the 2D-to-3D reconstruction problem.

In "style-based inverse kinematics" [Grochow et al., 2004], a scaled Gaussian process latent variable model (SGPLVM) is trained to build a likelihood function of human poses then used as a soft constraint with inverse kinematics problems, as well as 2D-3D point matching (shown in Fig. 2-7). They compare their method to the alternative of building a likelihood function by reducing the space of poses to the main components of a PCA, and modeling the prior using mixtures of Gaussians where the parameters of the mixture is found with expectation maximization (EM). They show the SGPLVM requires fewer dimensions while yielding better results. Their SGPLVM requires an active set of poses (either walking, running or throwing, etc), and does not scale well to large databases of various poses. In [Wei and Chai, 2011], a mixture of factor analyzers are used as a prior in a maximum a posteriori (MAP) framework, and reduces the reconstruction error by using a large database (2.8 million poses). Their results are comparable to *linear* component analysis, which in their experiment consists in computing the K nearest poses in the database, reducing them with a PCA, and optimizing within the newly found PCA space.

Instead of learning the prior on poses, it is possible to build them manually, by specifying soft constraints on the pose base on expectations of the poses. For instance, the work of [Lin et al., 2010] restricts itself to sitting poses, which allows to specify soft priors on the locations of body parts (e.g. pelvis near the chair). The also include the assumption of balance, and set joint limits based on the human anatomy. They solve the problem by using a combination of global stochastic optimization, with a final refinement using a BGFS-type local optimization.

The pose priors [Grochow et al., 2004, Wei and Chai, 2011] allow to directly obtain a full body pose including legs, etc as it encodes the correlation between the different body parts. These models are compatible with the line of action constraint I will describe in the next Chapter. The only difference is that unfeasible poses that include squash-and-stretch are hard to produce with these methods.

4.2 Sketching motion

Because it is such a challenge to lift a single 2D stickfigure into 3D without using data, the methods for sketching motion all rely on data. Sketching motion include sketching

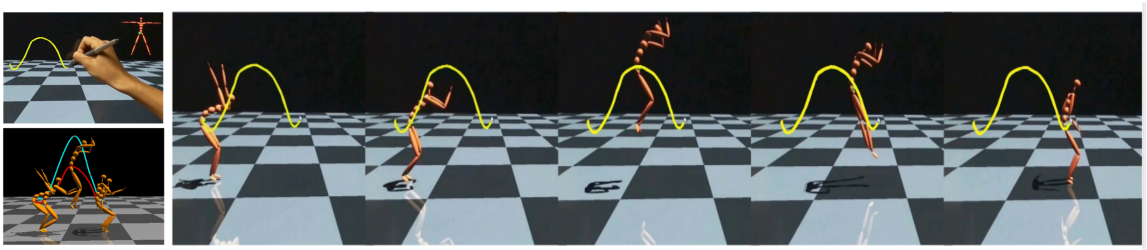

Figure 2-8: In Min et al. 2009, the user can sketch a point trajectory (left image), while the systen synthesizes a full 3D character motion, based on a data-base of pre-existing motions. In this case, we can see a data-base of a jumping sequence. The paper demonstrates the technique with 3 different motions: walking, jumping and golf swings. We can see a limitation of this approach for synthesizing motions that go beyond the data-base being considered.

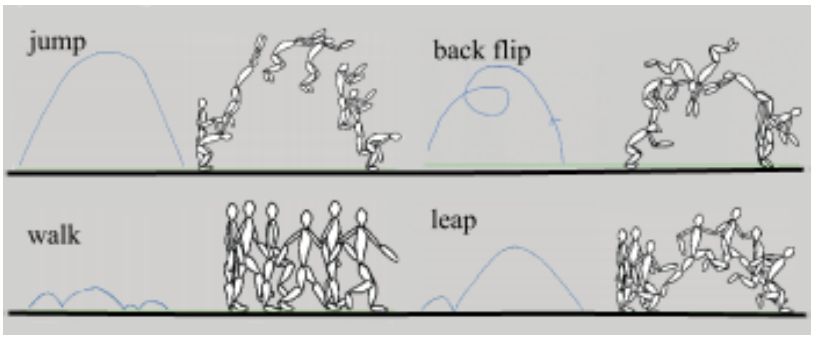

Figure 2-9: In Thorne et al. 2004, the user draws motion doodles to select pre-defined motion clips, whose timing is adjusted based on the speed of the stroke. This work allows cursive specification of motion (continuously gestured strokes, instead of discrete edits). Note that the gesture only controls the timing, and selects a pre-defined motion.

trajectories [Min et al., 2009, Yoo et al., 2014](shown in Fig. 2-8), and more abstract "doodles" [Thorne et al., 2004] (shown in Fig. 2-9).

Statistical priors. In [Grochow et al., 2004], the SGPLVM is applied to human motion data to reduce the poses (the angles and velocities) to a low dimensional embedding. The embedding provides a probability distribution function used as a prior on the space of 3D poses that is considered when solving for end-effector (IK) trajectory constraints. They model a space of discrete poses, which means that each pose is solved for individually. Others have looked at ways of modeling the motion sequence as a whole.

In [Min et al., 2009], a generative model of human motion is proposed where a set of motions are stacked (e.g. different walks) and a principal component analysis yields the vectors that captures most of the variance in the motion. The probability distribution

function prior over the whole motion is then constructed from the PCA components using a mixture of Gaussians where the parameters are found with expectation maximization. Having a generative model of the whole motion allows sparse position constraints, as well as end-effector trajectories (see Fig. 2-8)—to be solved in maximum a posteriori, space-time optimization framework. Similarly, though not used for sketching, Howe [Howe et al., 2000] tracks a 2D character in a monocular video reconstructs a 3D motion using a mixture of Gaussians learned from existing data. One of the limitations of modeling the space of motions is that it is built for a specific type motion (e.g. walking or running). Hence for such a method to be used in a general setting would require a way of knowing a priori which type of motion is likely to be performed.

Stitching pre-existing motion clips. A simple way of synthesizing movements matching a trajectory is to look into a database of existing movements for the closest matching clip to a user-drawn trajectory [Yoo et al., 2014]. The user selects a point in the body and sketches the trajectory. The system then computes the trajectories of the point in multiple viewpoints for each motion in the database, and selects the one that matches the most. Unfortunately, it is not always natural to draw exact trajectories for body parts. When people sketch motions, they often find natural to draw abstract paths.

The work of [Thorne et al., 2004] considers abstract paths (doodles) and associates them with pre-defined motion clips which are then stitched together using a motion graph. The stroke is segmented into different patterns that each contain characteristic shapes, as shown in Fig. 2-9. Each pattern is associated a pre-defined human motion. The motion synthesis uses motion graphs [Arikan and Forsyth, 2002, Kovar et al., 2002] to transition between the different clips. With this approach, the character's motion is not directly related to the shape of the drawn stroke—preventing from subtleties in the drawing to translate into subtleties in the motion. Moreover, it cannot be used directly with different characters, as it depends on a database of motions of specific character morphology.

Sketching motion in computer animation is a concept which seems to be making strong assumptions about the movement, and is geared towards novice users who need extremely simple tools to be able to animate. With such interfaces it is very easy to animate a character, but very hard to create a new and unique animation; not to mention

the need to animate various morphologies.

Sketching paths (to walk over). Several authors have proposed to control 3D animation by sketching 2D paths [Davis et al., 2008] embedded in a 3D world [Igarashi et al., 1998, Gleicher, 2001, Thorne et al., 2004]. These are typically drawn on the floor to control the global rigid 2D configuration of the character over time.

4.3 Human skeleton as an interface

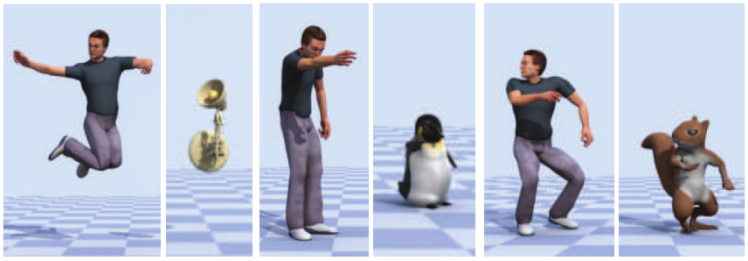

Figure 2-10: Human motion is used to animate simpler characters. Typically a calibration stage is required to establish a mapping. Image from [Yamane et al., 2010].

One of the most intuitive modes of communication is to simply act. In this setting, an actor's movement is directly captured. Traditional motion capture from optical [Kirk et al., 2005] or magnetic [O'Brien et al.,] sensors, require not only large spaces and expensive material, but also an expertise in recovering clean data which virtually always requires manual post-editing (e.g. by using the motion editing technique of [Lee and Shin, 1999]). In practice, human performance captured using these technologies are used for photo-realistic scenarios such as visual effects in movies, and realistic animations in games. It is not used in animated movies with characters performing expressive, artistic or physically unfeasible movements.

While using a full blown motion capture setup is cumbersome and expensive, the human data it acquires can be used to assist the inference of 3D human poses from partial pose information provided by cheaper and less cumbersome setups such as minimal marker sets [Chai and Hodgins, 2005], or depth imagery [Shotton et al., 2011] (the kinect).

In [Chai and Hodgins, 2005], a local model of poses is computed from the K nearest poses in the database, and used to infer on the missing marker positions. In [Shotton et al., 2011], decision trees learned from a large set of poses and their corresponding

depth image (in different views) are used to infer on the 3D skeleton pose from the single depth image. While these models inherently limit the space of possible movements humans can do, they can be combined with re-targeting methods to allow intuitively controlling simple characters by acting.

Figure 2-11: The user performs in front of the kinect to animate a character with different morphology.

There has been a recent interest in using 3D human skeletal information to control simpler, non-humanoid characters [Yamane et al., 2010, Chen et al., 2012, Seol et al., 2013, Rhodin et al., 2014]. In [Chen et al., 2012], the kinect is used to both scan an object, and animate it. Once the object is scanned (e.g. a chair), the human skeleton provided by the kinect is scaled and placed in the object in order to parametrize the surface vertices with a linear combination of the affine transformations of the K-nearest joints of the human skeleton. In [Seol et al., 2013], a set of features computed from the human skeleton (absolute and relative positions, angles, etc) are linearly mapped to a set of pre-defined character motions (the skeleton frames). The mapping is created with a calibration stage by acting in front of the kinect over the character motion. The coefficients of the mapping are found by minimizing a set of mapping error functions such as smoothness across frames. The works of [Yamane et al., 2010] and [Rhodin et al., 2014] are similar: to establish a mapping between a source (human) skeleton and target shape, the user first creates a set of corresponding poses (30 to 50 in [Yamane et al., 2010], and 4 to 8 in [Rhodin et al., 2014]). In [Yamane et al., 2010] a shared Gaussian process latent variable model is used to model the two space of poses (of each character) with the same latent space; yielding a passage from the human skeleton to the non-human skeleton via the latent space. In [Rhodin et al., 2014], the mapping is created by applying linear regression to the labeled poses yielding a linear mapping from the human skeleton to character's mesh.

5 Procedural and physically-based animation

The basic idea of procedural animation is to use some mathematical function to create the motion. In today's authoring software such as Blender [Foundation, 2015] and Maya [Autodesk, 2015], various functions exist to animate objects such Sinae and wave. Another possibility is to model the character's motion using the laws of physics. One of the main reasons for doing is for the ability for the character to realistically interact with its environment. The animation becomes the result of adding forces and torques to the joints, and simulating the discrete mechanical model, allowing external forces such as interactions with the environment to contribute to the character's movement. So far, this methodology is mostly used for realistic character animations. While there have been attempts at describing cartoon physics [Car, 1994], we have not yet used mathematical descriptions of cartoon physics.

Closest to this thesis are physically simulated characters controlled by the user, i.e. by manually adding forces [Laszlo et al., 2000]. Even simple characters yield highly complex dynamical systems which are hard for us (humans) to control. In fact, a popular game called Toribash [Toribash, 2015] lets players fight each other with a simulated character, where every second and muscle are controlled individually by the users in a turned-based fashion. Physics can also be useful for creating the *secondary motion* of characters (e.g. fat and muscles) that are kinematically animated [Hahn et al., 2012], or simulated [Liu et al., 2013].

If the character was not controlled, it would simply fall onto the ground like a ragdoll. For the character to perform certain tasks in a physically-simulated environment requires computing the necessary forces and torques. In other words, character motions are controlled dynamical systems.

There are mainly two formulations that consider the laws of mechanics for motion synthesis and editing: trajectory optimization which considers the motion as a whole (also known in computer graphics as space-time constraints) [Witkin and Kass, 1988, Cohen, 1992, Liu and Popović, 2002, Abe et al., 2004, Safonova et al., 2004, Liu et al., 2006, Mordatch et al., 2012, Borno et al., 2013, Wampler et al., 2014], and online controllers that continuously adapt and estimate the forces to add based on the current state of the character [Hodgins et al., 1995, Yin et al., 2007, Muico et al., 2009, Ye and Liu, 2010, Coros

et al., 2010, Wu and Popović, 2010, de Lasa et al., 2010]. Surely, there is an overlap between both approaches when some form of planning is involved. For instance, [Hämäläinen et al., 2014] proposes a predictive controller that solves for a temporal window ahead in time at each time step of the simulation.

Closest to this thesis is the space-time constraints formulation as it can resemble a keyframing setup where the animator specifies the motion directly by specifying shapes at different times, while the system searches for a physically plausible movement—while relaxing physical exactness when necessary.

Space-time constraints

Space-time constraints consists of specifying the character's shape, position and velocity at different places and time. Then an objective function suggesting how the character should perform the action (e.g. with minimal energy consumption) is minimized while seeking to satisfy mechanical relations (i.e. the laws of physics). The goal is then to optimize w.r.t. the space of possible trajectories for the one that satisfies both the constraints, physics and that minimizes the objective. While intellectually appealing, this type of formulation rapidly increases in dimensionality, and local optimization often results in a local minima.

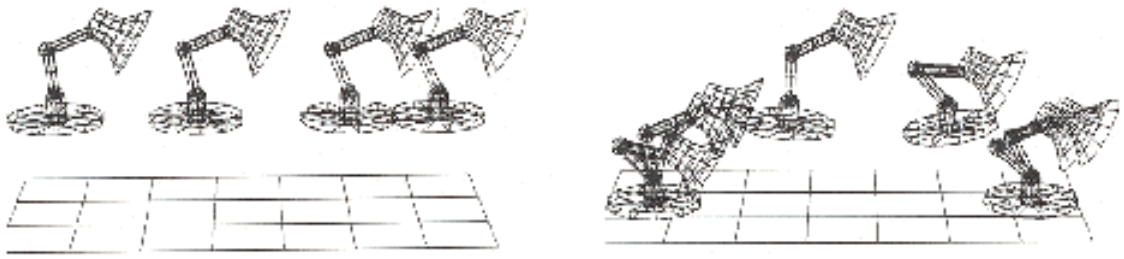

Figure 2-12: The space-time constraints [Witkin and Kass, 1988] looks for a physically feasible motion given a few user-provided constraints such as initial takeoff and landing positions. Trajectory optimization suffers from the curse of dimensionality, and local optimization often falls into local minima. It is not for any reason this motion is initialized in the air: it helps the optimization "find" the jumping motion.

The early work of [Witkin and Kass, 1988] considered a simple 2D character (a lamp holding a root at the tip and 3 angles) performing a jumping motion. Using local optimization requires initializing the motion close to the solution—in the case of Fig. 2-12, setting the initial frames in the air as to the optimization from resulting in a horizontal

translation motion. Instead of local optimization, Ngo [Ngo and Marks, 1993] uses a genetic global optimization algorithm and shows that different initializations lead to different motions. More recently, stochastic optimization was used along local time windows to solve for various 3D character motions [Borno et al., 2013].

Liu [Liu and Popović, 2002] uses simplified laws of physics such as center-of-mass acceleration equating gravity during flight stages, together with angular momentum profiles for ground stages. To avoid local minima, they initialize the motion with a simple motion using basic keyframes. In [Abe et al., 2004] COM momentum profiles are used to edit motions. Note, the center-of-mass profile has also been used to help animators keyframe motion by rendering a physically valid version of the COM trajectory to the animator [Shapiro and Lee, 2009], helping the animator edit its motion towards a more physically correct motion if desired.

The challenge with space-time constraints is the high-dimensionality of the problem causing local optimization to fall into local minima. Hence the methods of [Liu and Popović, 2002] and [Abe et al., 2004] initialize the motion with a close motion and then optimize to refine. While the physics constraints are simplified, the motion representation (character joints over time) remains high dimensional.

To improve the convergence of trajectory optimization, various works have reduced the dimensionality of human motion [Safonova et al., 2004, Chai and Hodgins, 2007]. It is true that many motions admit a low-dimensional structure. Walking can be described with a pose subspace [Safonova et al., 2004] with far fewer dimensions (7 or 9) compared to the full set of 54 joint angles. Spatio-temporal dimensionality reduction can represent the motion as a whole [Chai and Hodgins, 2007]. While dimensionality reduction improves convergence, most interesting motion includes sudden accelerations—something which low-dimensional models quickly lose.

The details we loose when reducing the dimensionality of the motion are at the extremities of our limbs: for straightforward mechanical reasons our hands and feet accelerate quickly as their motion is the result of larger muscles acting upon them at the other end of the chains. Hence foot contacts quickly become problematic. Additionally, the dynamics of contacts create discontinuities in the objective function landscape, which makes it hard to avoid local minima.

A recent work [Mordatch et al., 2012] has modeled the contacts as free variables in the

optimization problem converting them into continuous variables—allowing the solver to explore contact positions and timings. The character's motion is reduced to four position trajectories (2 hands and 2 feet connected to a root position) whose trajectories are represented with low-dimensional splines, and the final character configuration is recovered with inverse kinematics.

They show it is possible to initialize the motion with T-poses, and optimize locally. Once solved, the remaining parts of the body such as elbow and knees are recovered with inverse kinematics.

6 Style of animation

When keyframing, animators often start by drafting a rough motion, then towards the end of the process will add variations. Typically, overlaps or successions are added by delaying or shifting keyframes in time. This is done by copy-pasting the keyframes and shifting them in time. This process can be partially automated by warping within the parametric space of the keyframe splines [Neff and Fiume, 2003, Coleman et al., 2008]. Neff [Neff and Fiume, 2003] demonstrates anticipation, follow-through and successions with an arm motion defined by two keyframes. Coleman [Coleman et al., 2008] recovers keyframes from captured motion, and then applies the temporal warping technique of [Neff and Fiume, 2003] to create successions automatically. In [Wang et al., 2006], a filter based on a convolution kernel (the Laplacian of a gaussian) is used to alter the motion signal near the temporal boundaries (start and finish), automating anticipation and follow-through. In [Noble and Tang, 2006], the main direction of an action is computed to stretch the character as to exaggerate the action. In practice there is a limit to which these artistic effects can be fully automated. Most animation studios have tools for delaying and shifting keyframings in time, but in practice, it often requires manual intervention.

Different works decompose a motion into different modes and transfer the high frequencies to other movements as to alter their stylistic attributes [Unuma et al., 1995, Pullen and Bregler, 2002, Shapiro et al., 2006]. Pioneering work [Unuma et al., 1995] showed that removing high frequency bands computed with a fourier decomposition from one periodic motion and adding them to another could transfer stylistic attributes.

Similarly, [Shapiro et al., 2006] applies independent component analysis and takes the less meaningful components of a motion to add them to the components of another motion. For assisting in keyframing, [Pullen and Bregler, 2002] decomposes captured motion into Laplacian pyramids and use the lower bands to match to user-created keyframed motion and then "texture" (add details to) the motion by adding the higher frequency bands. Their scheme allows detecting correlations between body parts. For instance, certain hip movements are correlated with knee movements which allows to synthesize complex motions from simple keyframed movements.

Several works have modeled style and variations for transfer [Hsu et al., 2005], interpolation [Rose et al., 1998, Torresani et al., 2006] and slight extrapolation [Brand and Hertzmann, 2000, Lau et al., 2009, Ma et al., 2010]. Style and variations have different interpretations based on the modeling approach. Style can be be viewed as variations that are consistent across structurally similar motions [Brand and Hertzmann, 2000, Hsu et al., 2005]. For instance, a bipedal locomotion is a class of structurally similar motions, and variations in style—due to age, weight, mood, etc—are consistent and recognizable across different gaits.

In [Brand and Hertzmann, 2000], hidden markov models are used to extract high-level stylistic parameters from motion sequences. Their model seeks to isolate parameters that capture the variations from the basic structure of a motion category. They show results of parameters manipulating gender, weight distribution, grace, energy, and formal dance styles, which can be applied to different motion categories (walk, run, pirouette, etc). The work of [Hsu et al., 2005] translates styles of motion by establishing a correspondence between the frames of two similar motions of common structure (e.g. trot and walk), as to compute the difference to the common structure w.r.t. each motion and re-target the differences.

In the case of [Lau et al., 2009, Ma et al., 2010], styles are variations within a specific category of motion (e.g. variations in a specific gait such as walking). They take as input sequences of similar motions (e.g. similar walks) and model the variations using generative models. To synthesize new motions that preserve the variations in the example motion, [Lau et al., 2009] models the motion with a Dynamic Bayesian Network (DBN) and similarly [Ma et al., 2010] uses a network of latent variables explicitly modeling and learning the correlations between body parts (arms, legs and torso).

With this interpretation of style, one approach consists of interpolating user-labeled motions according to stylistic meta variables [Rose et al., 1998, Torresani et al., 2006]. For instance, a set of walks could be labelled according to two axes: happy/sad, strong/weak. Using radial basis functions [Rose et al., 1998], the system then interpolates between and extrapolates around (slightly) the set of labelled motions. In [Torresani et al., 2006], captured motion is labeled by a Laban movement analysis (LMA) expert. The motion synthesis is cast as a regression problem where the motion parameters are found by minimizing the error on the LMA effort of the labeled motions.

It is worth mentioning that cartoon motion has been captured from 2D videos and transferred to 3D characters [Bregler et al., 2002]. They track 2D markers and compute affine transformations that can be transferred to 3D character skeletons or cages for animation.

Note that some works consider style simply as characters performing different tasks [Grochow et al., 2004] (throwing, or walking). This later work was focused on ensuring the poses resulting from solving under-constrained inverse kinematics problems remained within a space of stylistically similar poses—in this case walking or throwing—by using non-linear dimensionality reduction (SGPLVM) and steering the poses towards the closest likely pose. The black areas in the low dimensional latent space (shown in Fig. 2-7 correspond to the *mean* pose when transformed back into the high dimensional space of the character's pose. The coloring (black-white) reflects the likelihood of the pose.

3 | Posing with a line of action

Figure 3-1: The line of action helps create more expressive and readable poses by directly specifying the overall flow of the pose with a single stroke.

- Published in *ACM TOG (in proceedings of SIGGRAPH Asia 2013)*.

In Chapter 2 we learned about several flexible methods to create poses (rigs, stickfigures, tangible devices, etc). The problem with these methods is that they do not provide control over high-level aesthetic and artistic aspects of the pose. It is very easy with these methods to violate the principles that make a pose expressive. An expressive pose is one that we directly understand: by looking at the pose, we easily know the intentions, actions or even state of mind of the character.

How to create expressive poses is an art well documented in guidebooks on drawing [Lee and Buscema, 1978, Blair, 1994, Hart, 1997, Brooks and Pilcher, 2001]. To facilitate the reading of a pose, artists try to provide it with an overall coherent flow. Flow is a principle of art and has to do with the way the viewer reads through the shape. A clear flow will allow the reader to more easily understand the pose, its intention, action or expression. To help control this high-level artistic aspect when drawing, artists use a tool called the line of action (LOA).

The line of action consists of *gesturing* a single stroke to directly specify the overall

flow of the pose, and only then to populate the body of the character around the stroke. This tool helps ensuring all the details of the drawing coherently align onto an aesthetically pleasing curve that facilitates the reading of the pose. In Fig. 3-2 we can see expressive poses created with one or two lines of action compared to ambiguous poses that hold ambiguous flows: a straight line (left) or too complex curves (right).

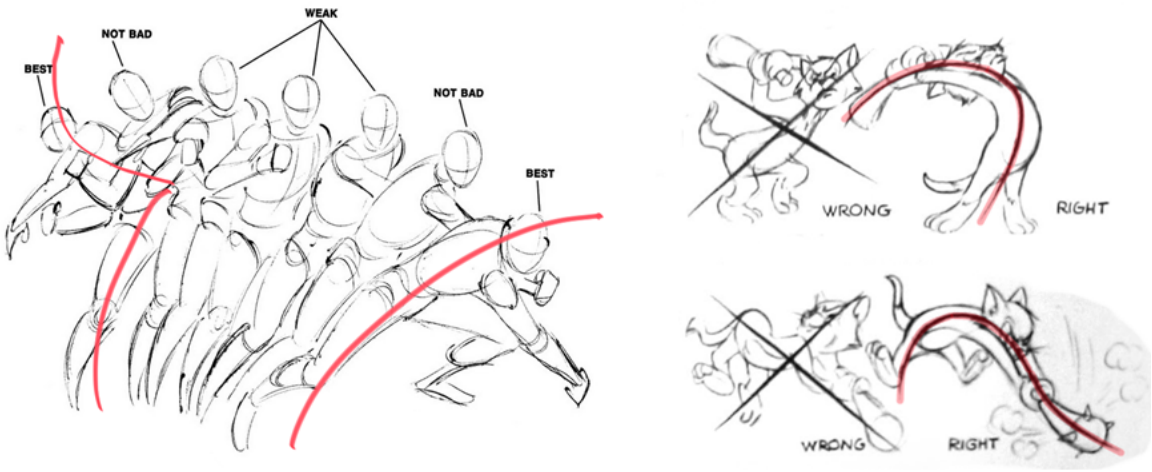

Figure 3-2: Poses with a clear line of action are more readable and expressive. By flowing through the pose, we can more easily understand the intention or action of the pose.

Flow, called movement in the principles of art [Pri, 2015], is the way the viewer will read through the pose (see Fig. 3-3). As an example, imagine starting from the head of the character and gazing (flowing) down to the feet. Different shapes of curves convey different emotions. For instance pointy features will convey tension, while smooth curves are more relaxed and feminine. Guidebooks on drawing recommend restricting the LOA to a family of simple curves. They recommend using C- and S-shaped curves. Providing the pose with a unique smooth curve will clarify the main expression of the character, which is also an important animation principle (simplicity and appeal). Using two strokes with a pointy feature as shown in Fig. 3-2 will add tension to the pose. The idea is to keep the LOA simple and use as few curves as necessary to convey the action or intention.

In this chapter, I will describe methods to pose a 3D character by drawing a 2D line of action. While the line of action is covered by many guidebooks and tutorials on drawing [Lee and Buscema, 1978, Blair, 1994, Hart, 1997, Brooks and Pilcher, 2001, Doucet, 2011], there is no line of action tool for 3D animation. The reason why is because we do not have a formal definition of the line of action that can be translated into mathematical

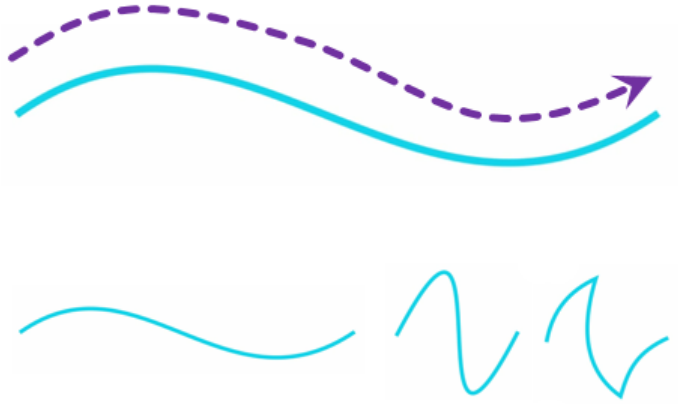

Figure 3-3: Artists use the terms flow when refereing to the act of reading through the LOA or pose of the character. The term rhythm refers to the type of patterns in the LOA's shape. Low curvature will have a relaxed feel and rhythm, while high curvature and sharp corners will add tension yielding a tensed rhythm. Note that in the principles of art [Pri, 2015], flow is called movement—a term which I believe would be too confusing in a thesis on motion.

terms. In other words, it is a challenge to write the character's shape as a function of the line of action. Hence the main contribution of this chapter is a definition of the LOA, formally relating the line to the character's body. This definition can then be used to formulate 3D posing as an optimization problem. Results of this system allow rapidly creating expressive poses such as the ones shown in Fig. 3-1, where we see a fashion model stance, two dance poses, a full body swing and a super hero on the run.

Technical overview

Posing a 3D character using a single 2D line of action is a highly ambiguous problem. Realizing a 3D posing tool based on this concept requires addressing a series of technical issues.

In the first section (Section 1), I will start by providing a way for users to confine their strokes to C- and S-shaped curves. A weak definition for this family of curves is that they hold 0 and 1 inflexion points respectively. To constrain the strokes to this family of curves, I provide a smoothing method based on fitting a low-dimensional parametric curve (a Hermite spline) to the drawn stroke. This can be removed for expert artists that master gesture.

Then I introduce a formal definition of the line of action. From this definition, I formulate an optimization problem to solve for a 3D character pose given a sketched line of

action. Essentially this problem can be viewed as a line matching problem, i.e. matching a set of bones to a drawn line. I will directly describe an algorithm to solve the problem assuming we know the correspondences between the body and the LOA, i.e. which bones are to map to which part of the line.

Because the sketched LOA provides only 2D information for the character and we need to solve for a 3D pose, the problem is under-constrained—several (solutions) poses can fulfill the line. Many techniques exist to further constrain the space of poses [Grochow et al., 2004, Lin et al., 2010, Wei and Chai, 2011] (discussed in Chapter 2). However, they confine the poses to specific tasks [Lin et al., 2010] or to realistic human poses [Grochow et al., 2004, Wei and Chai, 2011]. Unfortunately these methods will sacrifice the flexibility and prevent the method from creating exaggerated poses for arbitrary characters. To allow free-form poses for arbitrary characters, while constraining the space of solutions, I introduce a viewing plane constraint. This allows the animator to create unbalanced and mechanically impossible poses, but requires rotating the camera to deform in the depth dimension.

The correspondence between the body and the line of action is highly ambiguous. We need to know which bones are concerned by the line of action, and to which area of the line they map to. These problems are addressed in sections 4 and 5, where I provide tools to select the bones, and methods for computing the correspondences between the bones and the LOA.

Strokes notation The line of action will be denoted as a continuous curve $\mathbf{x}_{loa}(s)$. The most free-form shape for the LOA is simply the curve built from interpolating the recorded 2D samples of the stroke, i.e. $\mathbf{x}_{stroke}(s)$.To remove local variations in length for the LOA, I re-parameterize the input stroke $\mathbf{x}_{stroke}(s)$ to a constant length curve $\mathbf{x}_{cl}(s)$—meaning the space between each sample is of equal length. In other words, the stroke has constant velocity: $\|\frac{\partial \mathbf{x}_{cl}}{\partial s}(s)\| = c$. In the remaining of the Chapter, the LOA curve is always initialized with a constant length curve, i.e. $\mathbf{x}_{loa}(s) = \mathbf{x}_{cl}(s)$.

1 The line of action

The line of action's description often differs from one artist to the next. To formalize the concept, I provide a definition which, in my view, spans a wide range of cases. Artists rec-

ommend using a certain range of possible shapes for the LOA; namely C- and S-shaped curves. I will formalize this concept and provide a tool that helps users remain in this family of curves. Then I will introduce the definition of the line of action relating the character's body to the LOA. Note that the definition is general enough to be used with any shape of LOA and is not necessarily restricted to C- and S-shaped curves.

C- and S-shaped curves. In most textbooks on drawing, the LOA is restricted to the family of *C*- and *S*-shaped 2D curves. This restriction can be useful for two reasons: it constrains the poses to aesthetically pleasing shapes, and to physiologically plausible poses. C- and S-shaped curves are defined as having zero and one inflexion points respectively.

To reduce the variability of the line, I fit a cubic Hermite curve with two nodes (position and tangent per node). The positions are set each extremity of the input stroke $\mathbf{x}_{cl}(s)$. Then I optimize w.r.t. the tangents of the Hermite spline to best fit the samples of $\mathbf{x}_{cl}(s)$ with the Hermite spline yielding an optimal curve $x_{loa}(s)$. It is possible to increase the number of control points to provide more flexibility while smoothing the input curve $\mathbf{x}_{cl}(s)$. To do this we can subdivide curve by inserting nodes halfway between each node along the stroke and optimize for all the tangents and the middle node's positions' as well.

LOA definition: the line of action dictates the *shape*, in image space, of a subset of the character's skeleton (i.e. the bones). For instance, we often see the line setting the bones going from the head to a foot, as shown with the yellow bones in Fig.3-6.

For clarity of comprehension, it is useful to view the constrained bones in the character's body as forming a line (linear connected chains of bones)—which I will refer to as the *body line*. In other words, the line of action dictates the shape of the body line.

Body line. The body line is formed from the character's skeleton and will be denoted $\mathbf{x}(s)$. I assume the character has a skeletal subspace mapping the surface geometry via *skinning*, and the skeleton is kinematically linked with relative angles. My definition of the line of action could be used with other parameterization of shapes (e.g. cages [Sederberg and Parry, 1986]). However, they would require providing a skeletal function w.r.t.

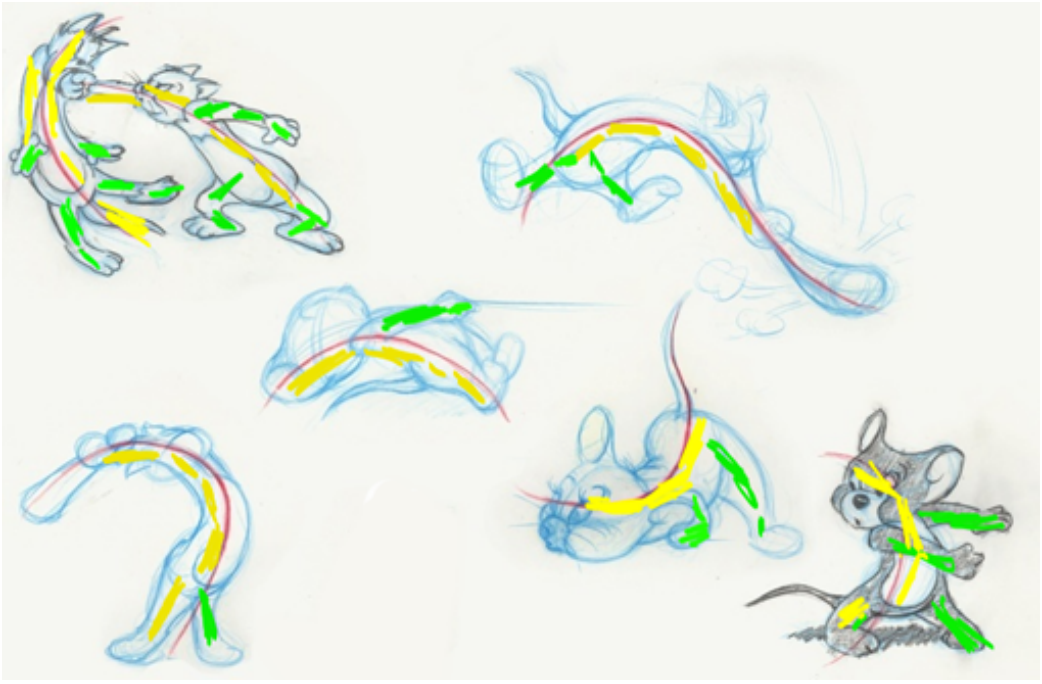

Figure 3-4: The line of action is a skeletal abstraction of the character's shape, drawn from a specific viewpoint; i.e. in image space. Scribbled in yellow are the bones I believe are constrained by the line of action, and in green are those I believe are not. We can see how the bones do not necessarily touch the line, but the shape (orientation) matches: see for example, the character's leg on the bottom left.

the new degrees of freedom (e.g. the skeleton as a function the cage nodes). The positions of the skeleton, in contrast, are direct functions of joint angles.

For the sake of clarity, we can use the same parameter s for both the LOA and body line. In Section 5, I describe methods for computing a mapping from the bone points to points on the LOA.

Let the position in screen space of the line of action be $\mathbf{x}_{loa}(s)$, the bones' positions $\mathbf{x}(s)$ and their position in screen space $\mathbf{P}\mathbf{x}(s)$, where $\mathbf{P}$ is a view and *perspective* projec-

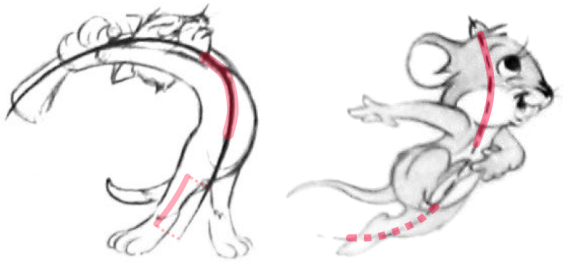

Figure 3-5: This figure shows how the line of action dictates the shape of the character which is not necessarily the position. We can see the leg of the character on the left does not touch the LOA but its shape matches. This is important for characters that hold offsets in their kinematic tree such as near the pelvis in humanoids. ©The Estate of Preston Blair, [Blair, 1994].

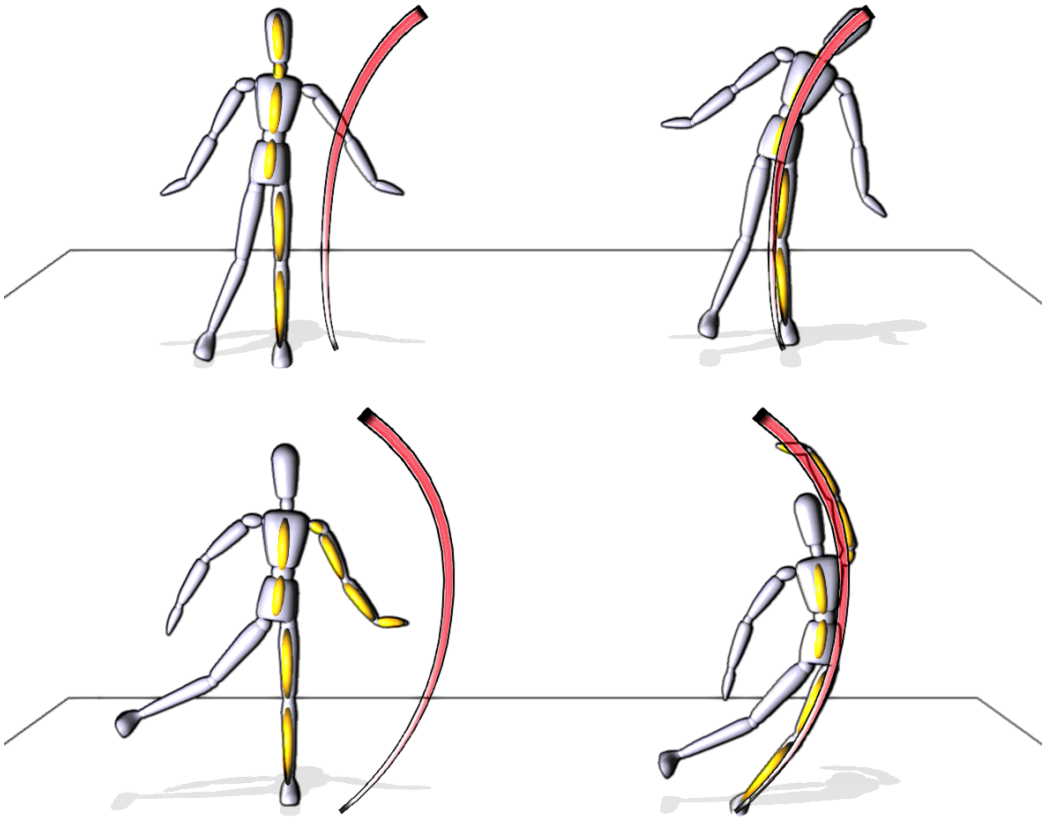

Figure 3-6: In yellow on the left, are the bones forming the body line that will be constrained and deformed according to the sketched line of action. On the right is the result after solving the LOA problem (3.1). In the top image, the body line goes from the head to a foot. In the second row it goes from a hand to a foot.

tion transformation in homogenous coordinates.

Following the definition of the line of action, the body line pose $\mathbf{x}(s)$ matching the shape of the line of action requires the first derivates be the same. To have both lines be as close as possible, I minimize the average distance between both lines resulting in:

$$\begin{aligned} \min_{\mathbf{x}_{root},\mathbf{Q}} \quad & \int_s \|\mathbf{P}\mathbf{x}(s) - \mathbf{x}_{loa}(s)\|^2 \; ds, \\ \text{subject to} \quad & \frac{\partial \mathbf{P}\mathbf{x}}{\partial s}(s) = \frac{\partial \mathbf{x}_{loa}}{\partial s}(s), \; \forall s, \end{aligned} \tag{3.1}$$

where $\mathbf{Q} = \{\mathbf{q}_0, ...\mathbf{q}_{N-1}\}$ is the set of bone rotations and $\mathbf{x}_{root}$ is the root of the skeleton.

The objective function minimizes the average distance in position. If the character has offsets in the kinematic tree such as between the pelvis and leg of a humanoid, then

it is crucial to have the shape (tangents) matching constraint, as shown by the example in Fig. 3-5.

The hard constraint over tangents can be met by using the lagrange multiplier method. I first experimented with the alternative of turning the constraint into a soft penalty with a larger weight for the tangents than for the positions. In both cases, this problem is under-constrained meaning many solutions (3D poses) exist for the same 2D LOA stroke. For instance, we can simply imagine a line drawn shorter than the character in screen space. It could mean the character has translated away from the camera, but it could also mean that he has crouched. In the next section (Section 2), I further constrain the possible solutions.

Note that this line of action concept cannot be directly realized with the *IK spline* tool in Maya [Autodesk, 2015] because of the *shape* term in problem 3.1. The IK spline snaps bones onto a spline. However, this does not allow skeletons that hold offsets in their kinematic tree to match in shape. With an IK spline, a humanoid's leg would have his leg snap to the LOA instead of having its shape matching the LOA as in Fig. 3-5.

2 Viewing plane constraint

What is interesting is how to reduce the space of the problem, without sacrificing the flexibility of the resulting tool. In other words, it should allow bending the character into exaggerated poses as well as work with arbitrary morphologies.

To reduce the size of the problem while allowing arbitrary bendings of the body parts, I make the assumption that the character is deformed from an initial pose, and I constrain the transformation to the viewing plane. This constraint can be met by construction simply by parameterizing the bone orientations we are solving for, with an axis-angle representation where the axis points in the viewing direction.

Hence, the 3D bone orientations q_j we were solving for in problem 3.1 boil down to a single angles θ_j. The corresponding axis is the camera's viewing direction projected onto the floor plane. The translations are also parameterized to be parallel to the viewing plane, *i.e.* along the 2-components u, v of the viewing plane. In the next section (Section 3), we solve the line matching problem (3.1) with respect to the θ_j and u, v.

The viewing plane constraint allows the user to freely exaggerate bending angles and

making unbalanced poses, adding drama to their figures—as often done by cartoonists. On the other hand, the user has to turn the camera in order to "edit depth"—while previous edits orthogonal to the viewing plane can be preserved, as illustrated in Fig.3-7 where the user rotates the camera to bend the knee. As we will see in the next Chapter (Chapter 5), the viewing plane constraint does not prevent the LOA from being used with arbitrary characters such as dragons.

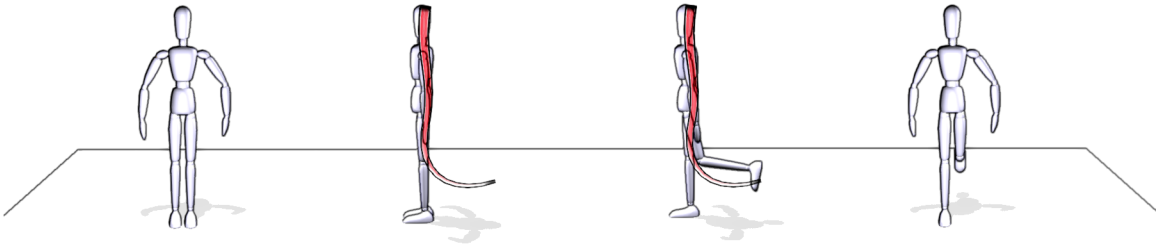

Figure 3-7: When solving for the character's pose to match the LOA, the bone rotations are constrained to the viewing plane. This constraint is met by construction: the rotations are parametrized as a single axis-angle component—reducing the joint angles to a single angle. The axis is the camera's viewing direction projected onto the floor plane. **From left to right** *is the initial pose, a side view with the line of action stroke, the result after optimization, and finally a frontal view of the resulting pose.*

3 Solving with non-linear programming

The viewing plane constraint reduces the angles of each joint j to a single unknown angle θ_j. The bone orientation is thus defined as: $q_j^{'} = q_j exp(\theta_j \ \vec{\mathbf{v}_{\mathbf{view}}})$, where $\vec{\mathbf{v}_{\mathbf{view}}}$ is the viewing direction and exp is the exponential map turning the angles $\theta_j \ \vec{\mathbf{v}_{\mathbf{view}}}$ into an orientation.

The viewing plane constraint also reduces the root translation to a space of planar translations $\mathbf{x}_{root}^{'} = \mathbf{x}_{root} + w_u\vec{u} + w_v\vec{v}$, where $\vec{u}$ and $\vec{v}$ are orthogonal vectors in the viewing plane.

We now solve problem 3.1 with respect to the reduced set of degrees of freedom: $\Theta = \theta_0, ..., \theta_{N-1}, w_u, w_v$. The shape constraint in problem 3.1 is a hard constraint that will be satisfied by adding a penalty to the objective function with a larger weight for tangents:

$$\min_{w_u, w_v, \boldsymbol{\Theta}} \quad \int_s w_x(s) \left\| \mathbf{P}_{\text{vp}}\mathbf{x}(s) - \mathbf{x}_{loa}(s) \right\|^2 + w_\partial(s) \left\| \frac{\partial \mathbf{P}_{\text{vp}}\mathbf{x}}{\partial s}(s) - \frac{\partial \mathbf{x}_{loa}}{\partial s}(s) \right\|^2 ds. \tag{3.2}$$

Solving this problem with local non-linear optimization requires the first derivatives for the objective function w.r.t. the angles Θ as well as w_u and w_v component. These are computed numerically with finite differences. When the problem is solved, we set the new bone orientation to the previous one $q_j \leftarrow q_j^{'}$, and the angle θ_j to 0. Similarly, we set the new value of $\mathbf{x}_{root}$ to $\mathbf{x}_{root}^{'}$ and both w_u and w_v to zero.

In practice, there are a number of issues when solving this problem with local optimization. The first one is that because we solve only for the angle, in other words for rigid bones, it helps to normalize the tangents in problem 3.2: without normalization, larger bones weigh more in the objective function which is also an issue mentioned in [Lin et al., 2010]. The second issue is the type of solver to use when numerically differentiating with respect to angles. I experimented with a BFGS solver from the library (DLIB [DLIB,]) but estimating the Hessian from the numerically computed gradient could lead the solver to diverge. It was more reliable to use gradient descent with small steps which is the solution used for the results shown in Section 6.

For the moment we assumed a single parameter s mapping both lines. Next, I describe how to compute a correspondence between points on the bones and points on the line of action. As in most cases, not every bone in the body is constrained by the line, it first requires selecting the body parts (Section 4). Once we know which bones are constrained, we can find for each point on the bones where it maps to on the LOA.

4 Selecting the bones

The line of action can be used to modify different sets of bones or using our terminology: different body lines. Sometimes the *body line* includes an arm, sometimes both legs, while in other cases it includes only the arms—as with *secondary lines* [Abling, 2012].

The weakest, most general definition of the body line is that of a subset of *connected* bones in the kinematic tree of the character. In my view the association of bones onto the line is mainly an artistic choice that should reflect the way it is done when drawing: by drawing body parts around the line. However, it would be time consuming to go through this process each time a line of action is sketched.

In this thesis, I investigated rapid ways of selecting the bones. The underlying idea is to have pre-defined sets of bones that are often used (e.g. head to foot, arms, torso,

etc). I investigated selecting the set of bones automatically, by taking the set closest to the drawn LOA. In practice, this approach is very confusing and it is much more practical to simply select the set of bones from a pre-defined tab (by pressing keys).

The body lines commonly used are often *maximal* and their graph *linear*. These assumptions lead to a small set of body lines that can be exhaustively searched for the closest one every time a LOA is drawn. The terms *connected*, *maximal* and *linear* are formally defined as follows:

Connected: The bones are children or parents of one another in the full kinematic tree. For instance, if the LOA goes from one foot to the other, it also includes the two lower and upper legs.

Maximal: Most body lines start and end at extremes of the full kinematic tree. Restricting our definition to maximal lines leads to a set of 10 possible kinematic chains.

Linear: Each node between two bones that are not the extremities are of degree 2. The extremity nodes are degree 1 nodes; they have only one parent or one child.

In the case of a humanoid, there are only 10 maximal chains: *head to left hand, head to left foot, head to right foot, head to right hand, left hand to left foot, left hand to right foot, left hand to right hand, left foot to right foot, right hand to left foot and right hand to right foot.*

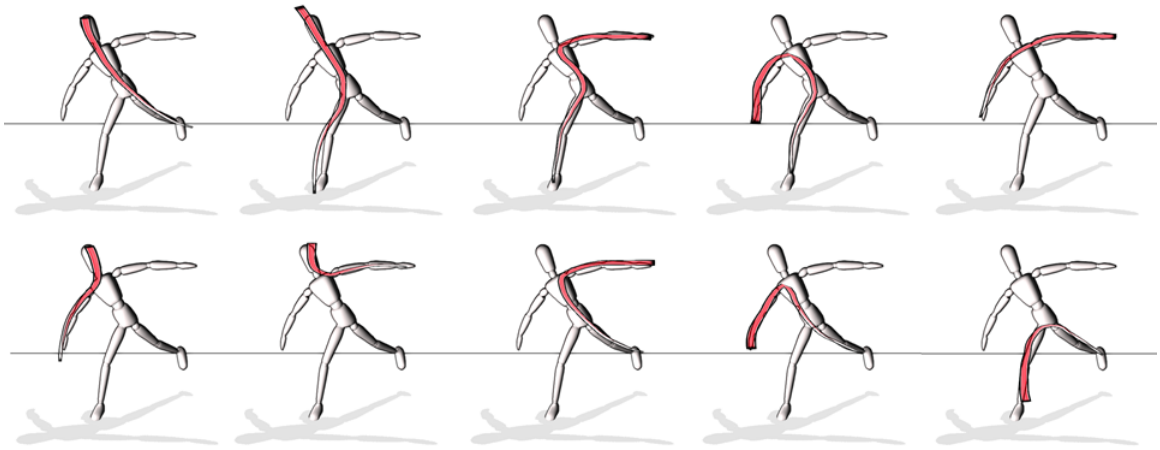

Figure 3-8: If we consider only linear maximal chains as body lines, than we only have 10 possibilities for humanoids. This can be used to automatically select the body line by choosing the one which is closest to the drawn LOA.

5 Mapping bone points to LOA points

Given the body line, we know which bones are constrained by the LOA. However, to match both lines, we need to where the bones map to on the line of action. In Section 1 where the line matching problem is introduced, we assumed a common parameterization s for both lines. Here we will compute for every bone point i^h, i^t (head and tail), a corresponding point on the line of action s_i^h, s_i^t.

Sometimes the line is drawn larger or shorter—globally or locally—than the body line. For example, Fig. 3-9 shows an S-shaped curve with a larger upper S for the upper body, and we can see there should be a jumps in the mapping between the bones in the upper body and the bones of the leg. To compute the mapping, I first experimented with the iterative closest point method where geometric information such as curvature can be used to direct the mapping towards areas that would be preferable—e.g. high curvature must not be covering a bone.

Iterative closest point (ICP) approach described bellow in Section 5.1, consists of estimating a first mapping from the initial pose and the LOA based on the closest point, to then solve for a first pose with this mapping (problem 3.2), and then re-compute the mapping and repeat the process until convergence.

The iterative closest point method can be very tricky to work with. The basic term we minimize for the mapping is the distance. However this alone can create drift and have the character slide as shown in Fig. 3-9. I added several terms reflecting different expectations of what should constitute a good mapping. For instance we can exploit features such as high curvature areas which are likely to be a corner in the skeletal structure; and are therefore not likely to be covered by a bone. We can expect more *flexibility* in certain areas of the mapping. For instance, we can assume the space between the upper body and lower body to be distant, allowing a jump between them, while we expect the head and foot to remain close to the tips of the line (and be less flexible).

The problem is that each of these different terms need to be fine-tuned and can have different behaviors depending on the pose and drawn LOA. In later chapters, I use a simpler method (described in Section 5.2) which consists of computing only the closest-point from a reference line that is first sketched on top of the body line, as shown in Fig. 3-10. Once computed this mapping can be stored and re-used for all subsequent

edits. Interestingly, with the reference line we can view the LOA problem(3.1) as a deformation transfer [Sumner and Popović, 2004] problem, where the transformation from the reference line to the LOA is transferred to reference skeleton pose to deform it.

5.1 Iterative closest point

The iterative closest point approach consists of computing the closest point on the sketched LOA to the body line point, and then performing an iteration of line matching using the algorithm described in Section 3—and repeat until convergence for both problems. The problem is that if we only optimize the closest point objective $E_{\mathbf{x}}$, the mapping can slide along the line as in fig. 3-9. Hence, we add soft constraints (priors) on the mapping $S = [s_0, ..., s_{M-1}]$ based on the expected positioning and connectivity of the skeleton's tree structure, while penalizing bones that would cover high curvature areas.

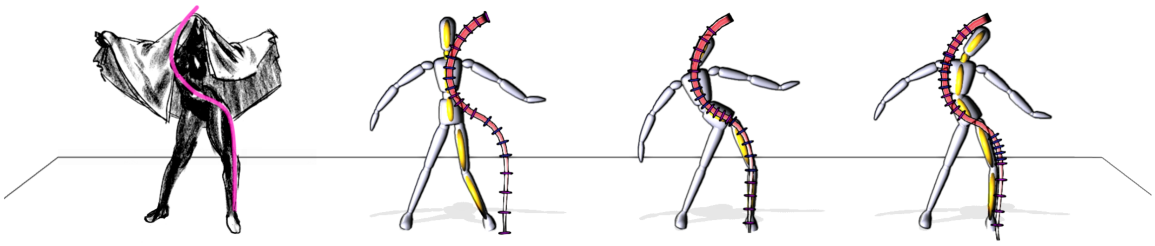

Figure 3-9: **Left:** *A classic S-shaped curve often used for a feminine stance. It is difficult to know a priori to which part of the line the bones should map to. The upper body seems to be mapping to the upper part of the line, while bottom part at the bottom tip of the line with an empty hole in the middle near the pelvis. In this thesis, I experimented with an ICP approach (Section 5.1) which computes the closest point mapping during online optimization of the pose. The three last images are: the initial pose, the resulting parameterization when using only the closest point objective, and the final image is the result when adding constraints based on prior expectations of the mapping (when solving problem (3.3)). Illustration ©Ben Jelter.*

In short, we can expect the bones to avoid covering areas on the LOA with *high curvature*, which translates into a penalty term $E_\kappa(w(s))$. To allow *jumps* in the solution (see Fig.3-9), the warping function is defined piecewise linear, each corresponding to a rigid bone interval on the body line. Assuming a skeletal tree structure, the associated mapping should preserve the parent-child relations established by the skeleton; bones should not overlap or go too far from one another—making a connectivity penalty $E_C(w(s))$ rel-

evant. The final energy function to minimize is as follows:

$$\min_{S} \int_{s} E_{\mathbf{x}} + E_{\kappa} + E_{C} \; ds \,, \tag{3.3}$$

$$E_{\mathbf{x}} = w_{\mathbf{x}}^{i} \left\| \mathbf{P}_{\text{vp}} \mathbf{x}_{b}^{i} - \mathbf{x}_{loa}(s_i) \right\|^2 ,$$

$$E_{\kappa} = w_{\kappa}^{i} \left\| \frac{\partial^2 \mathbf{x}_{loa}(s_i)}{\partial^2 s} \right\|^2 ,$$

$$E_{C} = w_{C}^{i} \left\| s_{i+1} - s_i \right\|^2 .$$

Note, that the results shown in this chapter use the ICP method, where at each step of the optimization (solving problem (3.2)), the optimal mapping is computed by solving problem (3.3). In subsequent chapters, the simpler uniform mapping described below is used and is computed only once at initialization.

5.2 Uniform mapping

Given a character in a reference pose, we can simply ask the user or the system designer to sketch a reference curve $x_{ref}(s)$ over the constrained set of bones in order to compute the mapping. For each bone position x_i, we compute the closest point s_i on the reference curve $x_{ref}(s)$:

$$s_i = \underset{s}{argmin} \| x_{ref}(s) - x_i \|. \tag{3.4}$$

This mapping is computed only once at initialization.

6 Results & Discussion

As a proof of concept, I evaluated my method with the task of reproducing static poses from photographs, cartoon images, and finally, example motions from video frames. The poses were then exported to Maya for interpolation and rendering. The line of action posing implementation is a stand-alone software that offers all the necessary tools for

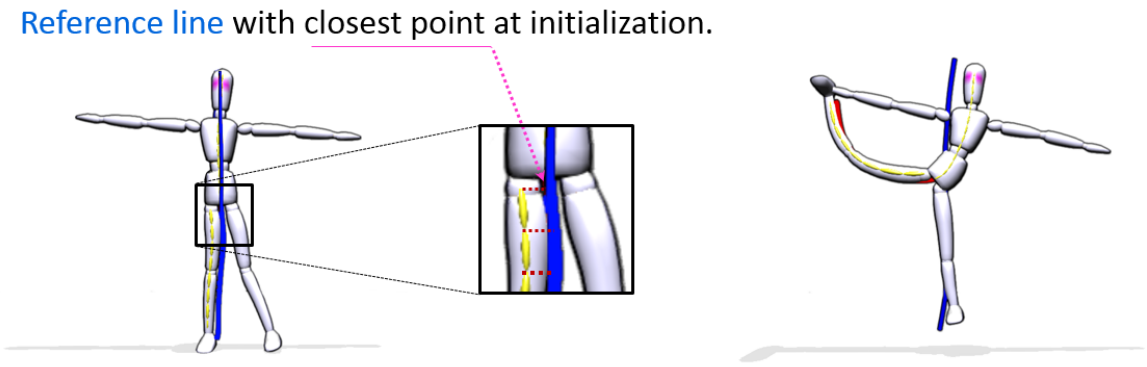

Figure 3-10: Use a reference shape to compute the closest point on the LOA to the bone points. In my implementation, the reference curve is used only at initialization to compute the mapping. Once the mapping is computed, we can freely edit the pose as shown on the right.

drawing multiple lines of action in multiple viewpoints, and computing the corresponding poses at interactive rates.

In my implementation, I solve problem (3.2) using gradient-based local optimization. At the start of each step, I solve problem (3.3) for an optimal mapping. In both cases I optimize until the gradient's length is smaller than a small criterion. The optimization is interactive and can be visualized on-line.

I made all the poses and animations myself using lines of action only and I am not to be considered an expert animator. For the evaluation, I measured the total number of gestures (strokes and rotations), as well as the actual time taken to generate a satisfactory result. Each time, I started from a neutral pose, and drew all the strokes necessary to complete the pose or sequence of poses. Then I asked a professional animator (with five years experience) to reproduce the same poses using FK/IK skeletal rigs in Maya. We measured the time required to finish the same task and summarized the evaluation in table 3.1.

What this evaluation demonstrates is that all the example poses could be generated within reasonable times. While this is not a complete comparison, we can extrapolate that a line of action tool can be used to produce rough poses and keyframes in a fraction of the time it would take with existing posing tools. The benefits seem especially spectacular when considering longer animation sequences like the *Muybridge* sequence—shown in Fig.3-16 and the accompanying video. It took a total of 6 min. to make using lines of action while it took 22 minutes to a professional artist using traditional 3D ani-

Figure	Num. Strokes & Rotations	Time LOA	Time Maya
Single poses			
Walk (Fig.3-17)	4 st. 1 r.	20 s.	90 s.
S shape (Fig.3-9)	3 st. 0 r.	20 s.	120 s.
Hero Punched (Fig.3-11)	6 st. 2 r.	90 s.	150 s.
Hero Punch (Fig.3-11)	7 st. 4 r.	2 m.	3 m. 30 s.
Animations			
Dancers (Fig.3-14)	6 st. 1 r.	30 s.	2 m. 45 s.
Cartoon swing (Fig.3-13)	7 st. 1 r.	45 s.	2 m. 30 s.
Head spring (Fig.3-16)	34 st. 1 r.	6 m.	22 m.

Table 3.1: Number of strokes, camera rotations and time taken using lines of action v.s. using 3D FK/IK widgets in Maya.

mation tools in Maya.

The main reason is probably the fact that multiple bones get quickly aligned onto the LOA and fewer edits are required. But the foremost benefit of this method is the fact that beginners *could* actually create expressive poses using a line of action, while there is no way in Maya for beginners to easily create poses that exhibit an aesthetic and coherent flow with their shape. The line of action is particularly useful to create dramatic poses that convey full body expressions more clearly.

In practice I found that most poses were made by sketching 3 strokes: one to specify each half of the full body (right leg, left leg), then a third to adjust the arms using a *secondary line*. Moreover, it is worth noting that once a pose is created, variations of the same pose can be easily tried-out by drawing slight variations of the initial line. This explains why making a basic walking pose (Fig. 3-17) takes only 20 seconds while making a dramatic punch destined for comics like the one in Fig.3-11 takes minutes; we tend to redraw the same stroke over previous ones to perfect the pose and make it more dramatic.

Note that concurrently to my work, Oztireli et al. have considered lines of action [Öztireli et al., 2013] for posing character. They focused on sketching the shape of partial body parts. In contrast, the methods in this Chapter allow sketching full body lines with offsets in the kinematic tree—such as at the hips of a human. Their focus is on supporting large deformations, while I focus on formalizing the LOA and providing an interface to edit character shapes with LOAs that cover the full body.

6.1 Limitations and future work

One of the main limitations of this work is that it only solves for the orientations of the bones and not their lengths. In contrast, expressive animation requires devices such as squash and stretch to exaggerate actions and intentions. This problem is solved in Chapter 5.

The ICP approach for computing the mapping between the bone points to the LOA points involves many terms in the optimization problem (5). Each of them has to be fine tuned and it can be hard to find parameters that work across different poses and lines. For this reason, I proposed an alternative and simpler mapping based on first sketching a reference line—allowing to compute the mapping without ambiguity. In subsequent Chapters, this simpler method is used.

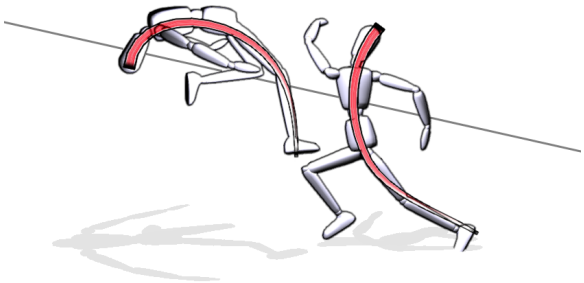

Figure 3-11: Inspired by comics, these characters are posed fighting. In each of these poses, we can see the overall body expression controlled by a line of action.

A limitation of the viewing plane constraint is that bones can no longer twist. I explored the possibility of letting the bones rotate around their own axis (the direction they point to). However this does not provide an intuitive visual representation of twist, nor does it help specify the twist reliably. For example, in Fig. 3-11 the upper body is rotated around the spine by sketching a secondary line for the arms while rotating the camera. In the next chapter, I solve this problem with method borrowed from the drawing literature: a technique called the “Two Cans” technique.

I mainly explored rapid selection of body parts to be constrained by the line (predefined bone sets). The way the line of action is used in drawing is by populating the body around the line. It could be interesting to investigate ways of allowing the user to roughly sketch the body parts along the line while the system computes the mappings and poses the character.

One of the advantages of the optimization problem (3.1) is that we can easily add extra constraints. For instance, when the line intersects the floor, we can insert a foot or hand contact constraint to the problem (3.1). This was used in Figures 3-13, 3-14 and 3-17 for foot contacts. See also next chapter for more examples.

The definition of the line of action is quite general and can be applied directly to other morphologies like quadrupeds. For example, a line of action can be formed from the head going through the spine and finishing at a foot. Secondary lines can go from one leg to the other. Similarly, we can think about refining other shapes this way. For instance hands and multiple characters can be connected with a single stroke. A hand could be posed by drawing a line from a finger to the thumb as shown in Fig. 3-12.

Figure 3-12: The line of action can be used to pose multiple characters and other shapes such as hands.

7 Conclusion

The line of action helps animators create more expressive and readable poses by directly specifying the overall flow of the character with a single stroke. To achieve this, I had to

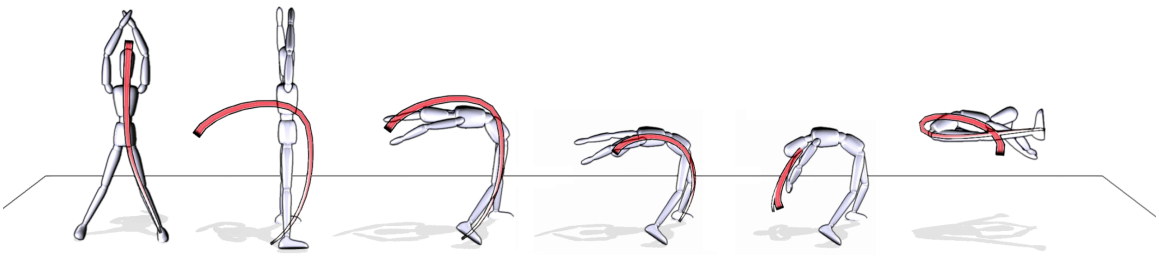

Figure 3-13: The main poses of a cartoon character swinging a bat take only 8 strokes and 1 camera rotation after the first stroke. The overall process took less than a minute compared to several minutes with traditional 3D animation tools.

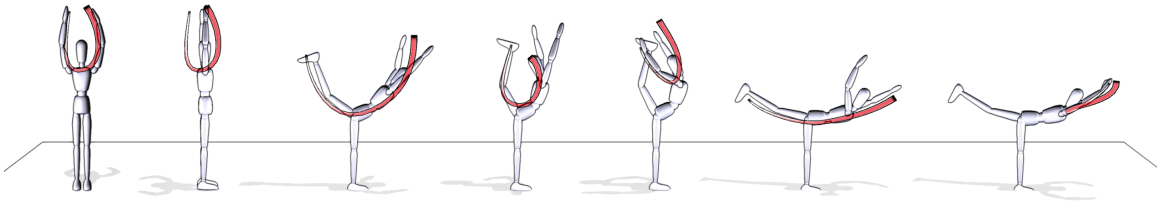

Figure 3-14: Dancers seek aesthetic and LOA-friendly poses by shaping their body into aesthetically pleasing curved shapes.

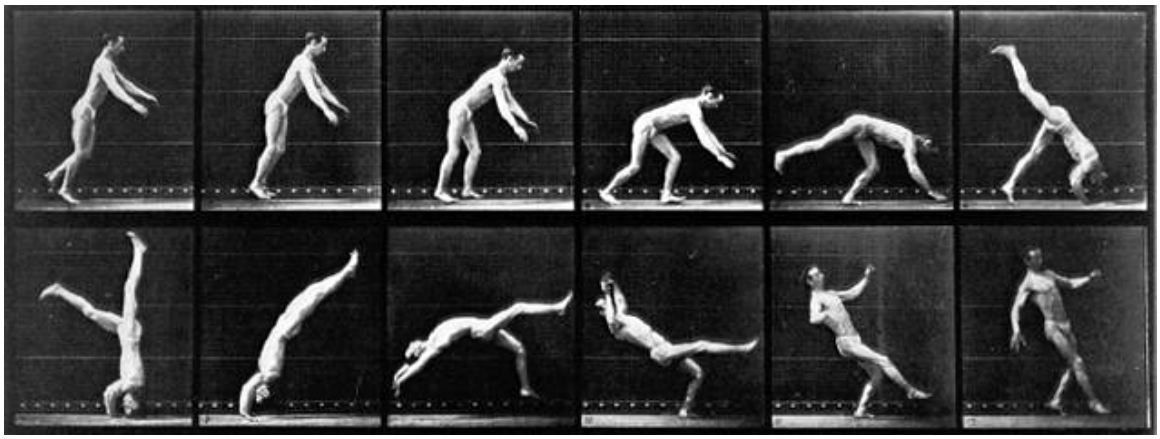

Figure 3-15: Head spring sequence captured by Muybridge.

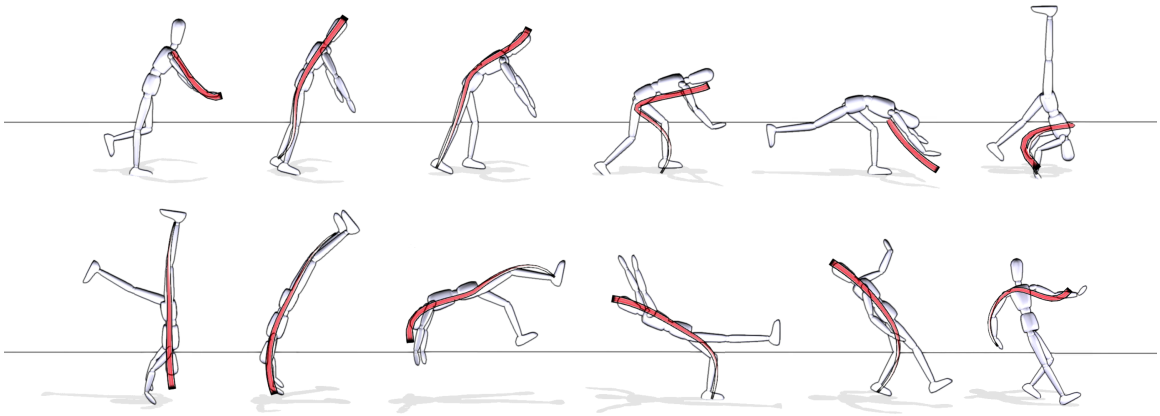

Figure 3-16: The keyframes of a head spring motion captured by Muybridge (see original pictures 3-15) were reproduced in 6 minutes by drawing lines of action from a single viewpoint. For each pose, I show the last line of action that was drawn.

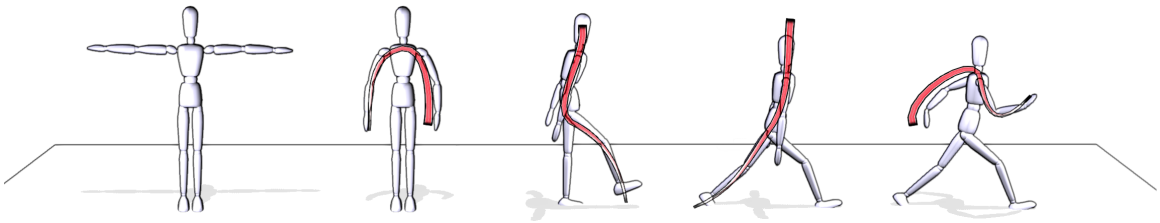

Figure 3-17: Starting from a neutral pose, only 4 strokes are required to get started with a walking animation. Note that the so-called secondary lines are used here to place the arms. Enabling the use of 2 Hermite curves for representing the LOA was essential to capture the poses of the two arms in a single gesture, due to the need for a high curvature point.

formalize the line of action concept. Then using this mathematical definition of the LOA, I formulated 3D posing as an optimization problem. To solve this problem, I provided solutions to several of the ambiguities related to this 2D abstraction of the body; namely depth and correspondence ambiguities. Results show the rapid creation of expressive poses ranging from elegant fashion stances to exaggerated cartoons in action. I briefly compared this method with traditional authoring software and found it was significantly faster for coarse pose design—especially for *expressive* poses that are challenging to create using traditional posing techniques.

The line of action determines the flow of a pose at a specific instant in time. Artists sometimes control the overall flow of a motion with a single continuous 2D line motion abstracting the motion of a character whose body parts are dynamically made to join and depart from the dynamic line. In the next chapter I explore this concept together with the possibility of simulating the line of action as an elastic—providing a continuous LOA motion from few strokes.

4 | Adding dynamics to sketch-based character animations

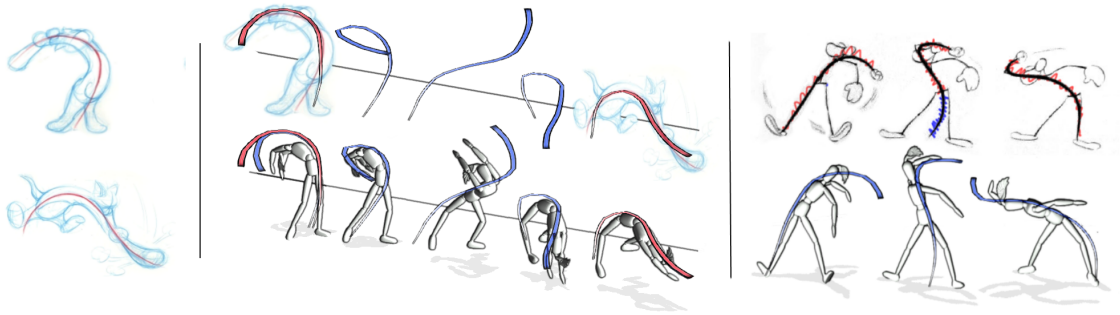

Figure 4-1: Artists typically sketch lines of action to convey motion. In the artist sketch on the left, we can almost "feel" the dynamics of the line, how it seems to be storing potential energy. Perhaps we are perceiving or inferring on the character's main muscle contractions. Inspired by these drawings, I devised a physics-based model for the line's motion which allows to unleash the line physically and to interpolate between sketched lines of action, while exhibiting anticipation and follow through (middle sequence). The moving line may require changing body parts dynamically, which my 3D character motion synthesis can take into account (right figure). See also Fig.4-3 and Fig.4-5. Cartoon images ©The Estate of Preston Blair, [Blair, 1994].

Summary

The animations in the previous chapter were produced by interpolating the 3D configuration of different poses created with LOA edits. Instead, this chapter begins work on directly interpolating 2D lines of action and then matching a 3D character motion to the dynamic LOA. Note that the 2D LOA interpolation is slightly improved in the next chapter. In this chapter, I focus on the idea of animating the LOA as an elastic, as well as elastically interpolating between keyframes. In the next chapter, I will provide new

sketch-driven motion models for the dynamic line (of action) that allow drafting its full motion from a single stroke.

Having a continuous line motion abstracting the motion of a multi-limbed character requires dynamically changing body parts being constrained by the DLOA. I provide a first semi-automatic method were the user annotates the moments where the body parts snap or detach from the moving line while the system propagates to the rest of the motion the common subset of body attachments. Note that the actual matching of a 3D skeleton to the 2D line is improved in the next chapter, and that the body swapping technique is compatible with the improved matching algorithm.

Lastly, I introduce the twist cans in this chapter allowing the user to reliably specify 3D twist around the line of action. The twist cans are also compatible with the improved line matching algorithm of the next chapter (Chap.5).

- Published *in proceedings of the Symposium on Sketch-Based Interfaces and Modeling (SBIM) 2015.*

1 Introduction

In this chapter, I explore the inherent dynamics that we often feel in hand-drawn line of action sketches. For example, if we look at the cartoons on the left of Fig. 4-1, we can almost sense the motion from the drawn line of action alone—the same way we can visualize motion between panels in comics. Indeed, lines of action introduced in Chapter 3 are often drawn by artists *"to make the pose more dynamic"*. The question is how?

One explanation is the clarity of the pose—facilitated by an easily readable flow—allows the viewer to more easily understand the action or intention. Another related reason is that the line of action often *extends through the action* of the character. To explain this concept, consider Fig. 4-1 where the character Tom is swinging his bat on the floor. In this case, the line of action starts from the collision on the floor and extends through the body of the character. Another example is when a cartoon character is punching (see Fig. 1-3 in Chapter 3), the line of action starts at the fist and extends through the body of the character. This has the effect of making our eye flow towards the main action of the scene (character punching)—helping us imagine the motion.

It is true that we humans have strong priors on humanoid motion from our life-

long experience watching others perform different actions. These priors allow us to infer full motion from simple abstract drawings such as lines of action. Following this path would naturally lead to data-driven techniques for synthesizing motion from these sketches. However, in the case of cartoon motions that may be exaggerated or stylistic, the example-based solution may not be sufficient.

In this chapter, I take a simple view of the line's dynamics, that will allow physically interpolating between different strokes, while exhibiting desirable motion features such as anticipation and follow-through; two important animation principles [Lasseter, 1987]. By viewing the line's shape as storing potential energy with respect to a target shape, we can derive a simple elastic model for the line's motion. This approach allows the user to specify keystrokes as the target shapes over time, while an adjustment method ensures the physically-simulated line matches the user-drawn strokes over time.

A physically-simulated line leads to new questions about the nature of the line's connection to the body over time. When the line can move freely, like an elastic, how should the body be driven by the line? Even before computers were widely available, animators designed continuous line motions as to ensure a coherent flow throughout the motion, as shown in Fig.4-4. If we pay close attention to the character's body in Fig.4-4, we can see different parts joining and departing from the line.

Understanding how these should be triggered is a challenge. It could be a purely artistic choice. In this work, I take a first step into automating these dynamic shifts in constrained bones, and provide a method that can solve for character motion matching the line while smoothly taking into account temporally discontinuous bone constraints. The resulting work flow requires the user setting the constrained bones at key moments (red squares in Fig.4-4) by sketching over a character in a T pose. The system then automatically solves for the common subset of bones of constrained bones and smoothly interpolates the parts that are not being constrained.

In short, this chapter holds two contributions:

- A physically-based line of action interpolation method.
- A dynamic skeletal line matching algorithm that transitions between discontinuously constrained bones allowing multi-legged 3D figures to be animated with a single moving 2D line.

To realize this physically-based sketching approach, I separate the problem into two sub-problems. The first is the creation of the 2D line movement by sketching and physically-interpolating strokes (Section 2). The second is a 3D motion synthesis, that solves for the character's skeletal motion as to match the 2D line's motion while taking into account discontinuous bone constraints (Section 3).

2 Physically-based line of action

2.1 Stroke representation

In this section, I describe the LOA's discrete representation, derive an elastic model for its motion, and describe how to ensure continuity across key-strokes specified by the user. When the user draws a stroke, the system records a set of 2D samples in screen space, which is denoted $\mathbf{x}_{stroke}(s)$. Simulating discrete elastic models on the cartesian coordinates of line (essentially a mass-spring system on the stroke) leads to spurious motions. One possibility would be to use a triangle mesh embedding. My first experiments consisted in decomposing the stroke into polar coordinates where the LOA curve $x_{loa}(s)$ is represented with a set of local angles γ_j and lengths l_j.

To transfer the recorded stroke $x_{stroke}(s)$ into the coordinates of a piecewise rigid chain, I first perform a constant length parametrization of the curve. Then I estimate the length of the chain's nodes $l = \int_s \frac{1}{N} \frac{\partial c_{stroke}(s)}{\partial s} \, ds$, which is the same for each rigid element j, given a number of elements N (10 in this experiment).

With forward simulations, the placement of the root position x_{root} of the chain will have an impact on the dynamics. Different placements such as the bottom tip, or at the middle of the stroke, will result in different motions. In these experiments I focused on reflecting humanoid dynamics and therefore placed the root at the center of the stroke $\mathbf{x}_{root} = x_{stroke}(0.5)$ with two chains going down to each tip of the drawn stroke.

Hence, the chain's root is set at the center of the curve $\mathbf{x}_{root} = x_{stroke}(0.5)$, and we fit the angles θ_j to match the stroke $x_{stroke}(s)$ in screen space:

$$\min_{\theta_j} \sum_{j}^{N} \left\| \mathbf{x}_{stroke}(s_j) - \mathbf{x}_{loa}(s_j) \right\|^2 , \tag{4.1}$$

where $\mathbf{x}_{loa}(s_j)$ is computed with the linked hierarchy of angles θ_j, and lengths $l_j =$

$l \; \forall j$.

Note that to simply interpolate the 2D strokes, we interpolate the angles and lengths separately using Hermite cubic splines, and then reconstruct the curve via forward kinematics, resulting in a dynamic line of action $\mathbf{x}_{dloa}(s,t)$.

2.2 Physically-based stroke interpolation

In this section, I propose an interpolation method based on on a physics-inspired analogy. This will allow generating rich motion between keyframes with few strokes. The idea is to estimate the internal tension or potential energy stored in the line of action (e.g. Fig.4-2). We may have this feeling due to our perception of the muscle contractions along the line. To realize this concept, I propose to view the LOA as an elastic where the animator can unleash its potential energy into momentum.

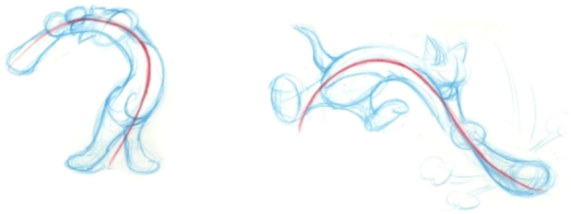

Figure 4-2: In this example, we can almost "feel" the dynamics of the line, how it seems to be storing potential energy. Perhaps these shapes trigger our perception of the character's main muscle contractions. Inspired by this artist drawing, I devised a physics-based model for the line's motion which allows to unleash the line physically, and to physically-interpolate between consecutive keystrokes. See Fig.4-3 for more details. ©The Estate of Preston Blair.

To achieve a given behavior for the elastic LOA, we must know what forces to add. This necessarily involves the character's intention, which is hard to infer from a single stroke. As in early physics-based controllers (e.g. [Hodgins et al., 1995]) we can define forces based on a target shape (by defininining an elastic potential energy as the difference in angles between both strokes). In other words, the animator could draw the target shape which will define the forces to add to the line.

This would only allow forward simulations. I explored the idea of sketching keyframes, and having the elastic LOA pass through each line while exhibiting desirable animation features such as anticipation and follow-through. The line starts at the first keyframe and and "shoots" for the next. When close enough, it changes it the next. To ensure matching

the keyframes, and prevent drift from accumulating I use a re-targeting method at each keyframe to warp the simulated frames as to have the closest simulated frame land onto the keyframe.

The simulation starts at a keyframe $t_{i=0} = t_k$, and moves forward over a set of frames $t_i = \{t_0 = t_k, t_1, t_2, ..., t_{M-1} = t_{k+1}\}$. To have the simulated frames hit the next keyframe at time t_{k+1}, we start by finding the closest *simulated* frame:

$$t_i^* = \underset{t_i}{\operatorname{argmin}} \sum_{j}^{N} \|\theta(t_i) - \theta(t_{k+1})\|^2. \tag{4.2}$$

Once we have the closest matching frame t_i^*, we compute a correction for the frames before t_i^*. This is done by measuring the offset to the next keyframe (t_{k+1}), and interpolating between 0 and the offset for all the frames between, resulting in the correction: $\Delta\theta(t_i) = (1-t)\,(\theta(t_{k+1} - \theta(t_i^*))$, which is applied to all the frames before $\theta(t_i^*)$.

In practice, I use $M = 40$ frames for the simulation between the keyframes. I found that simulating the root position of the lines x_{root} rarely produced adequate behavior. Hence for the root, I simply interpolate the position $\mathbf{x}_{root}$ between keyframes.

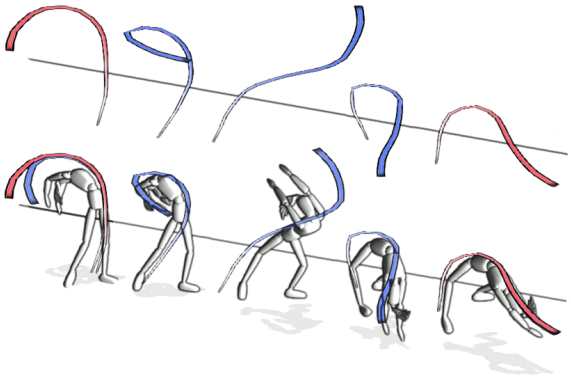

Figure 4-3: The lines in blue are frames of the physically interpolated line of action, which moves between two lines of action in red—drawn by the artist. Note how the motion exhibits anticipation and follow through, also shown in the video. The dynamic posing algorithm in the second row, ensures the character matches the line's shape in between the keyframes of the line.

t_i	time of frame i
t_k	time of keyframe k
q_j	local orientation of bone j
Q_j	global orientation of bone j
x_{root}	root of the kinematic tree
$x(s)$	interpolated position of a kinematic chain
x_j	position of bone j
$*(t)$	interpolated trajectory of d.o.f. *
$\Omega(t_k)$	set of bones constrained at keyframe t_k
$\Omega(t)$	$\Omega(t) = \Omega(t_k) \cap \Omega(t_{k+1})$ for a time $t \in]t_k, t_{k+1}[$

Table 4.1: Notation.

3 Synthesizing 3D character motion with dynamic bone shifts

In this section, I describe a first method for matching a 3D character skeleton to a moving 2D line. This approach is a direct extension of the static posing method described in the previous chapter to the dynamic case. Unfortunately, applying the method from the previous chapter sequentially to each frame of the animation leads to inconsistencies between consecutive frames. Alternatively, solving only for the keyframes of the physics-LOA and interpolating the 3D skeleton configurations will not have the skeleton matching the motion of the physics LOA between the keyframes.

In this section, I solve the LOA matching problem over space and time and use various smoothing schemes to ensure continuity of the motion—while matching the dynamic LOA. Then, I describe how to drive the motion of multi-legged characters with a single moving 2D line. The character's body parts are made to match the line over time. In my approach, the user annotates the body attachments at different keyframes and the system fills the attachments to the frames between the keyframes with the common subset of bone attachments (from the keyframe annotations).

Given the line of action definition in Chapter 3, the following should hold: at any instant in time t, a set of bones in the character's body should match the shape of the line—the shape being first derivatives or tangents. To have the character be close to the moving LOA, I proceed as in Chapter 3 and minimize the average distance between both lines:

$$\min_{\mathbf{x}_{root}(t),\mathbf{Q}(t)} \quad \int_{s\in\Omega(t)} \|\mathbf{P}\mathbf{x}(s,t) - \mathbf{x}_{loa}(s,t)\|^2 \; ds,$$

$$\text{subject to} \quad \frac{\partial \mathbf{P}\mathbf{x}}{\partial s}(s,t) = \frac{\partial \mathbf{x}_{loa}}{\partial s}(s,t), \; \forall s \in \Omega(t),$$

where $\mathbf{x}_{root}(t)$ is the root trajectory, $\mathbf{Q}(t) = \mathbf{q}_0(t), ...\mathbf{q}_{N-1}(t)$ the set of orientations trajectories of the constrained bones $\Omega(t)$ over time, and $\mathbf{P}$ is view and perspective projection matrix in homogenous coordinates.

Unfortunately, applying the method described in Chapter 3 sequentially to each frame of the animation is problematic. The convergence between consecutive frames can be inconsistent, leading to visual jumps in the animation. (Also, it does not take into account dynamic constraints for the set of bones being attached to the line, as in Fig. 4-5.)

To obtain a smooth motion, I formulate the dynamic matching problem as a space-time optimization problem, minimizing a temporal smoothness objective E_s with a (soft) shape matching constraint E_{loa}, assuming a *dynamic set of bones* $\Omega(t)$ to be described bellow:

$$\min_{\mathbf{x}_{root}(t),\mathbf{Q}(t)} \quad \int_t E_{\dot{x}}(t) + E_{loa}(t) \, dt. \tag{4.3}$$

Dynamic bone sets. The dynamic set of constrained bones is denoted $\Omega(t)$. We can observe (see Fig. 4-5) that changes in the set of bones $\Omega(t)$ happen at key moments, and that between these key moments, what is constrained is the common subset of bones. That is, assuming these changes occur at the keyframes, between the keyframes we have $\Omega(t) = \Omega(t_k) \cap \Omega(t_{k+1})$ for a time $t \in]t_k, t_{k+1}[$.

Smoothness. The smoothness term penalizes the difference between consecutive frames t_i. However, we do not wish to smooth out the motion of all the bones, but only those that are constrained by the line:

$$E_{\dot{x}}(t) = \sum_{j\in\Omega(t)} \|log(\dot{\mathbf{q}}_j(t))\|^2 + \|\dot{\mathbf{x}}_{root}(t)\|^2, \tag{4.4}$$

where $\dot{\mathbf{q}}_j(t) = \mathbf{q}_j^{-1}(t_{i+1})\mathbf{q}_j(t_i)$, log converts an orientation into a 3D axis-angle vector,

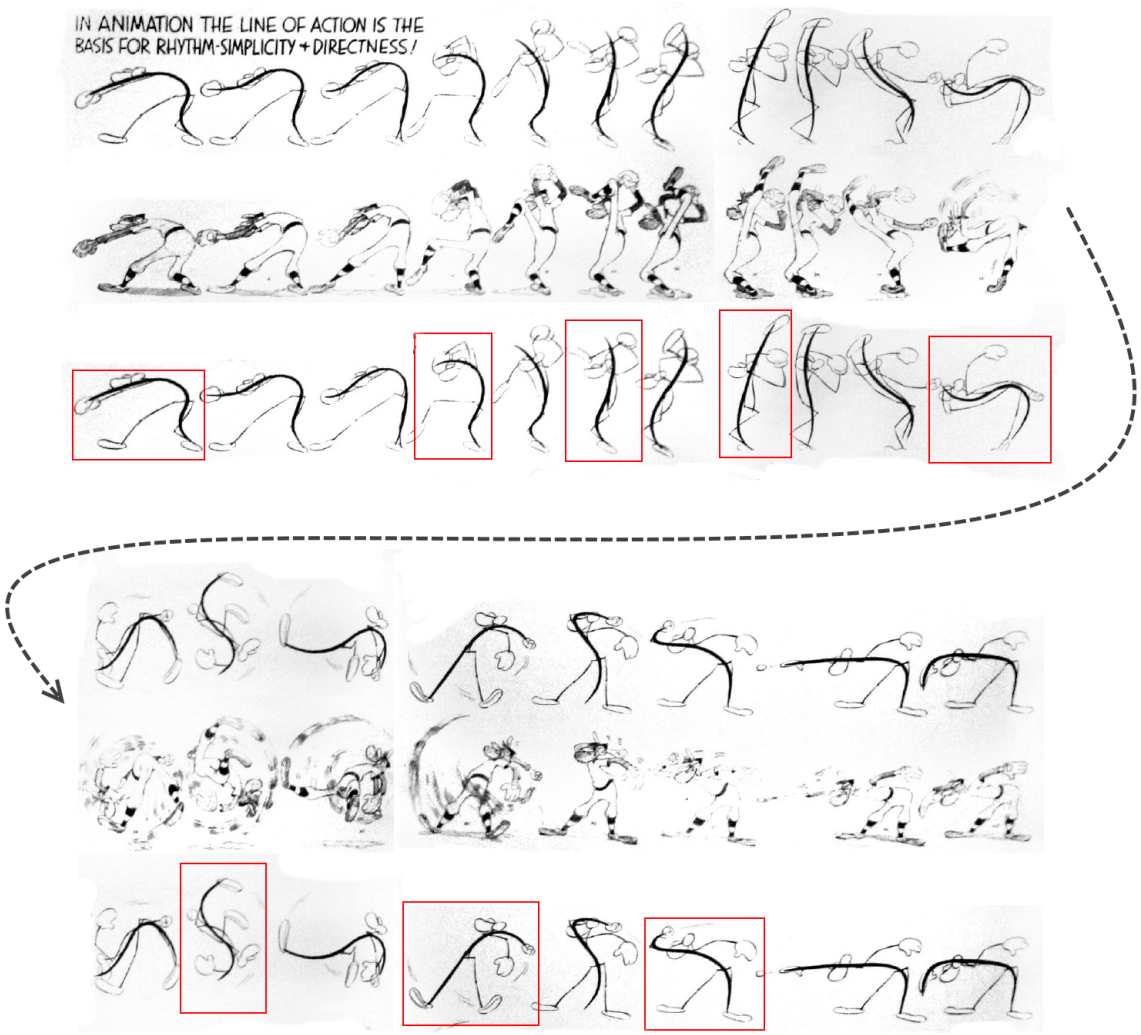

Figure 4-4: The animator designs a fluent line motion, and different parts of the body are attached to the moving line over time. In terms of body attachments, I hypothesize that the artist seeks to maximize the line's fulfillment (with the body), while respecting basic principles of articulated motion such as balance. In this chapter, I take a first step and observe that transitions happen at key moments (red squares), and that between the key moments, it is the common subset of bones from the key moments that remain constrained. Figure from the book Cartoon Animation [Blair, 1994], © The Estate of Preston Blair.

and $\dot{\mathbf{x}}_{root}(t)$ is the time derivative of the root trajectory.

Shape-matching constraint is turned into a soft constraint, resulting in the energy function:

$$E_{loa}(t) = \int_{s \in \Omega(t)} w_{\mathbf{x}} E_x(s,t) + w_{\partial} E_{\partial}(s,t)$$

$$E_x(s,t) = \|\mathbf{P}\mathbf{x}(s,t) - \mathbf{x}_{dloa}(s,t)\|^2$$

$$E_{\partial}(s,t) = \left\| \frac{\partial \mathbf{P}\mathbf{x}}{\partial s}(s,t) - \frac{\partial \mathbf{x}_{dloa}}{\partial s}(s,t) \right\|^2$$

where s is the space that covers each constrained bones in $\Omega(t)$.

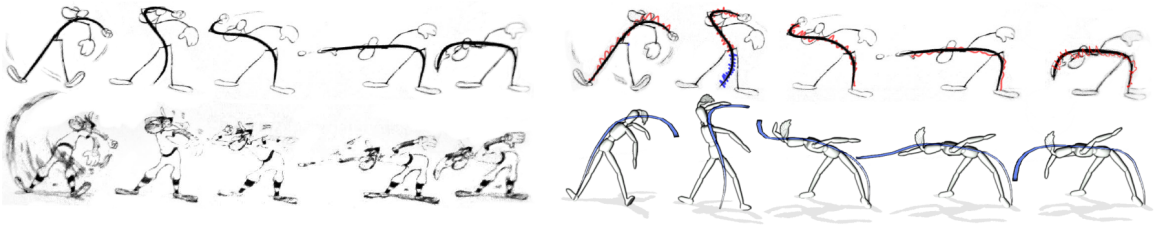

Figure 4-5: I scribbled in red the parts of the line that are constraining bones, and in blue the parts that are not. The motion synthesis presented in this chapter allows the line to transition between the right leg and the left leg. Example taken from the book Cartoon Animation [Blair, 1994], ©*The Estate of Preston Blair.*

Staging and solving for motion. In chapter 3, the problem is reduced by constraining the bone rotations to a single angle, where the angle rotates the bone in the viewing direction of the camera. It is true that many motions meant to be watched do happen in a plane nearly parallel to the viewing plane. However, the staging of the animation, can be at an angle from the viewing plane. More generally, I will refer to the plane in which the animation lies as the *action plane*, and the user is allowed to rotate it.

I solve for the angles controlling an axis-angle rotation in a direction $\vec{\mathbf{n}}$ (normal to action plane), and angle θ_j, for every bone j. The bone orientation of the character is thus defined as: $q_j^{'} = q_j exp(\theta_j \, \vec{\mathbf{n}})$. When the problem is solved, I set the new bone orientation to the previous one $q_j \leftarrow q_j^{'}$, and the angle θ_j to 0. The root position is parametrized as $\mathbf{x}_{root}^{'} = \mathbf{x}_{root} + w_u \vec{u} + w_v \vec{v}$ where $\vec{u}$ and $\vec{v}$ are orthogonal vectors in the plane. I optimize with respect to w_u and w_v and when finished, set the new value of $\mathbf{x}_{root}$ to $\mathbf{x}_{root}^{'}$.

The physically-based line of action will hold rich motion between keyframes. Solving for the character's pose only at the keyframes and interpolating the configuration of the character (the orientations and root) does not make the character match the line be-

tween keyframes. Moreover, I found that solving for each frame between the keyframes separately leads to inconsistencies between frames—even when using smoothing.

To obtain a closer match that remains smooth, I use a simple form of motion reduction. I use Hermite spline interpolation and optimize over this reduced set of dofs. In this experiment, I use one node for each keyframe stroke, and subdivide once between keyframes resulting in $2N - 1$ nodes, where N is the number of stroke keyframes. This scheme provides the extra liberty necessary to match the physically-based line of action, while remaining smooth.

Each frame is initialized with an initial pose, I solve (4.3) over the whole set of frames using local non-linear optimization. In the next section, I detail how to add 3D twist and contact constraints.

3.1 Additional 3D constraints

In the previous section, I solved the problem for a planar 3D character motion. With this approach it is quite difficult to represent and control aspects of the 3D motion such as 3D twist or foot contacts. In this section I extend the space-time constraints framework with additional 3D constraints for twist and contacts, which can be manipulated by the user using traditional 3D widgets.

This is done by introducing new terms to the optimization problem (4.3) reflecting twist along the line, which I denote $E_{twist}(t)$, and $E_{\perp}$ for contacts. Let me first describe the "Two Cans" drawing technique that inspire the twist specifications around the LOA.

The two cans drawing technique. 2D skeletal lines can hardly specify 3D twist. In hand drawings, the twist of characters is often established explicitly in the early stages of the drawing; that is, in the abstract form. Annotations such as lines or can-shape primitives shown in Fig. 4-6 are drawn over a line of action to establish the orientation. It is worth noting how the cans in Fig. 4-6 rarely conflict with the line's shape serving only as an extra dimension refinement.

The user controls twist widgets along the line to control a single twist angle. In other words, they do not affect the shape of the line, but only the orientation of the character. The twist widgets are attached to the line and assigned the orientation of the stroke. The twist angle is interpolated over time along the keystrokes with linear interpolation.

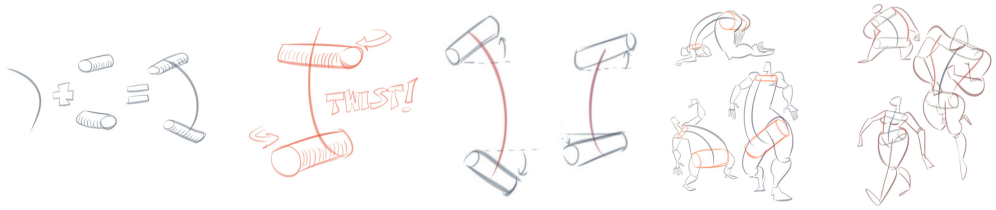

Figure 4-6: The Two Cans Technique is a drawing tool often combined with the line of action to help specify the twist and relative depth of the character. I use this visual handle to twist the dynamic lines in various ways. The user manipulates the cans to twist the body part over the space-time curve. Illustration by ©Krishna M. Sadasivam, [Krishna M., 2012].

The twist term. Each *twist can* constrains the orientation of a single bone, and all constraints χ are added into a penalty:

$$E_{twist}(t) = \sum_{j \in \chi} \|\mathbf{Q}_j^{-1}(t)\mathbf{q}_j^{can}(t)\|^2, \tag{4.5}$$

where $\mathbf{Q}_j(t)$ is the orientation of the bone j, which is not to be confused with the local bone orientation $q_j(t)$.

The contacts terms. The contacts are 3D rigid constraints for a bone (position and orientation). For instance, the foot has its position and orientation constrained by a contact widget. The user manipulates these traditional 3D widgets in the scene, and the character's pose seeks to satisfy them while satisfying the moving line's shape constraint. All the contacts are summed into a penalty:

$$E_{\perp}(t) = \sum_{j \in \Upsilon} w_x \|\mathbf{x}_j(t) - \mathbf{x}_j^{\perp}(t)\|^2 + w_q \|\mathbf{Q}_j^{-1}(t)\mathbf{q}_j^{\perp}(t)\|^2, \tag{4.6}$$

where Υ is the set of contacts, and $\mathbf{x}_j^{\perp}(t), \mathbf{q}_j^{\perp}(t)$ are the position and orientation trajectories of the contacts.

Putting it all together. These additional constraints may not lie on the *action plane*, as they are defined in 3D, i.e. twists, and 6D (position and orientation) end effector constraints. Hence, the final matching procedure is broken down into two steps: I first initialize the motion and solve for planar motion (Section 3), then I add the 3D constraints—the terms (4.5) and (4.6)—to solve for full 3D bone orientations.

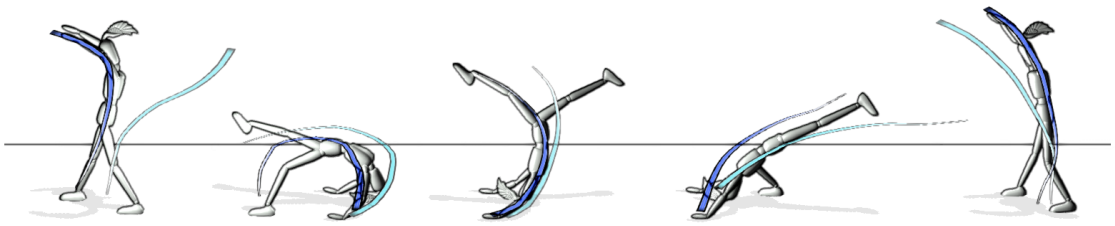

Figure 4-7: The backwalk in gymnastics is an example of a motion that holds a continuous line of action throughout the animation. The first pose has the right leg constrained, then the left leg for the 3 subsequent poses, and finally the right leg again at the last pose. The space-time optimization formulation takes into account contacts such as for the hands and foot. The user specifies these manually as in traditional animation software.

4 Results and discussion

I implemented these methods in a self-contained program. The sketch-based interface allows quickly drawing strokes and creating dynamic lines of action (Section 2). The user specifies the bones that are to be constrained at key moments by selecting from a set of pre-defined bone sets. Our system then automatically creates a smooth 3D motion that satisfies the discontinuous constraints (Section 3). The user can also edit secondary parts of the character that are not concerned by the main line by drawing strokes (as in Chapter 3), and can edit contact and twist widgets over time (Section 3.1).

I used the interface myself to generate the figures in the paper and the animations in the accompanying video. I used the physics interpolation method (Section 2) to create Fig. 4-3. Although it is possible to create interesting motions, it requires fine tuning the stiffness parameters of the angles over time as to obtain the right behavior. Once the parameters are set, it is only a matter of sketching a few strokes to create different versions of the motion.

I asked a few users to test this method in order to obtain feedback. I observed that users found the idea of sketching keystrokes to specify the dynamics not necessarily intuitive. Some would have preferred drawing strokes as forces, or to grab the line in order to wobble it as an elastic. These are interesting insights into how physics simulations could be used interactively to create movements.

I then tested this method for dealing with dynamic bone constraints (Section 3). Note that in these results, the line's motion was produced with basic geometric interpolation of the strokes (no physics). The animation in Fig. 4-5 requires setting a set of bones at a first

key moment for the right leg, and setting a second set of bones—holding the right leg—at another key-moment. In my implementation, the user selects the body parts from a set of pre-defined bone sets by scribbling a stroke over a set of bones. For the moment, it does not allow re-producing the long sequence in Fig. 4-4, where for instance, individual bones lie in the middle of the stroke. I used the same procedure to create Fig. 4-7—which transitions from the right leg, to the left leg, and back to the left leg. Note that a few additional results in the video show these capacities, while satisfying 3D contact constraints (Section 3.1).

5 Limitations and future work

In this chapter, I explored the idea of using a physically-simulated line of action that would mimic the action of the character's internal forces. The experiments showed it could produce desirable features such as anticipation and follow-through. However, I found that different users could expect different behaviors for the line, and I observed that fine tuning the stiffness parameters over time to obtain a given behavior was quite long. I believe this area could be improved with some training of the parameters, for example from cartoon videos or artist-provided examples.

While the sketching allows squashing and stretching the strokes, the 3D motion synthesis algorithm (the method introduced Chapter 3 and extended to time with smoothness in this chapter) solves only for piecewise rigid motion, and therefore does not allow squashing and stretching the character. Additionally, the solver requires a fair amount of smoothing, realized by adding a smoothness term and by representing the motion with a low-dimensional spline interpolation scheme—which in consequence, prevents from an exact match between the character and the line. This solver could be improved and allow squash and stretch—which is done in the next chapter (Section 5).

I made a first step into allowing dynamic transitions between body parts fulfilling the line over time. For the moment, the user specifies the constraints at key moments, but we could foresee a future where these transitions could be inferred automatically. For instance, we could approach the problem from a different direction: by using a controller for a simulated 3D character, and have the character seek to maximize the fulfillment of the moving line. Information from the character's 3D dynamics, such as balance and

contacts could be used to determine when the body parts should join or depart from the line.

The concept is demonstrated only with articulated humanoid characters, but the methods are applicable to other morphologies such as quadrupeds. In the future, it would be intersecting to investigate ways of fulfilling the motion with more abstract matters such as fluids that dynamically change topology.

6 Conclusion

By viewing the line of action as an elastic rod, the animator can create energetic movements for 3D characters by sketching fewer strokes. The dynamic line can change body parts over the coarse of its motion. To realize this, I provide a 3D motion synthesis procedure that supports smoothly changing body parts that are being driven by the elastic line over time. While the physically-based interpolation method requires fine-tuning stiffness parameters, it was able to reproduce cartoon motions. In particular, we demonstrated its effectiveness for producing anticipation and follow-through.

The line of action from Chapter 3 helps create more expressive and readable poses, but quality animation involves more than a small sequence of expressive poses. To create appealing motions, it requires coordinating motion over time. Something which the line of action provides no direct control over. In this chapter, I made a first step towards providing control over coordinated motion by adding physics to keyframed lines of action. In the next chapter, I introduce a sketch-based abstraction called the *space-time curve* that allows drafting a full coordinated motion with a single stroke.

5 | Space-time sketching of character animation

Summary

The line of action is mainly a static abstraction of motion and previous chapters focused on the idea of keyframing lines of action. This chapter introduces a new sketch-based abstraction of character motion allowing animators to draft a full coordinated movement from a single stroke. It also introduces the concept of motion composition allowing to layer different motions on top of one another. These different sketches translate into a dynamic line of action driving the character's motion. The way the character is matched to the moving line is improved (upon the previous chapter) with a fast and exact match that allows squash and stretch of the character's bones. This chapter also holds an improved LOA interpolation technique.

- Published in *ACM TOG (in proceedings of SIGGRAPH 2015*).

1 Introduction

Quality character animation requires more than a few expressive poses, it requires the careful coordination of the character's shape over time. However, in the main approach to free-form motion design (i.e. keyframing), the animator is always focused on the shape at a specific instant in time—preventing the animator from directly controlling the overall coordination of the motion. Hence, achieving quality results with the standard keyframing approach remains beyond the ability of unskilled users and time consuming for skilled artists.

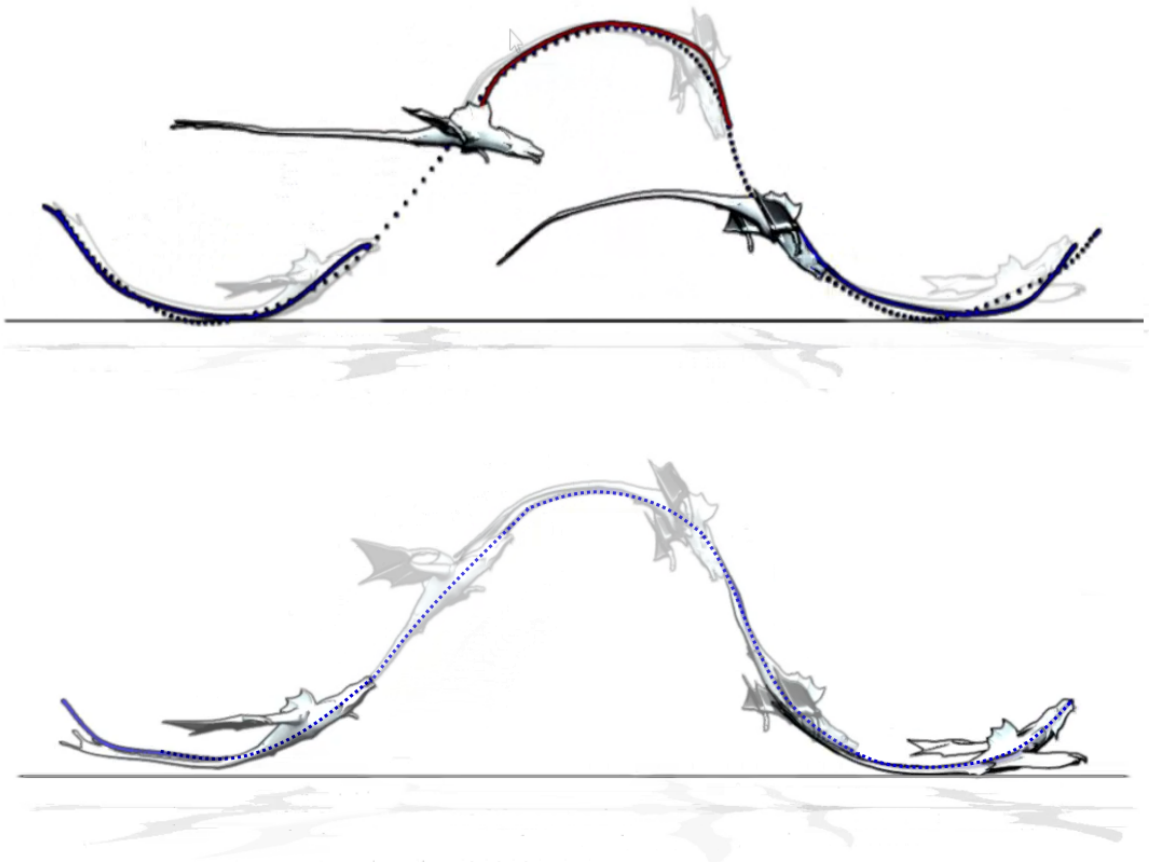

Figure 5-1: In the first image on the top, the character is animated with a trajectory for the root path, along the conventional "point-to-point" interpolation of the three keyframes (lines of action with the pose rendered in solid). This show how it is challenging to create path-following styles of movements with current interpolation methods. This animation requires a minimum of 5 keyframes with timing adjustments for a low quality path-following motion (see video). On the second image we see our approach where the animator simply draws a path to fluently animate the character as to follow the path. Later in this chapter (Section 3.2), sketched paths are composed on one another to sculpt complex motions such as a flying locomotion.

In this chapter, I introduce a novel space-time sketching abstraction enabling an animator to draft a full coordinated movement—that includes shape deformation over time—by sketching a single stroke called the space-time curve (STC). Further strokes (LOAs) can be used to progressively refine the resulting animation. While strokes have been used in the past to specify both temporal and spatial iso-values of motion—with static lines of action (LOA) serving as shape abstraction at a given time as well as trajectories describing the successive positions over time of a single point—*space-time sketching* was never used to define animations. In my approach, the user is allowed to control both

the shape and trajectory of a character by sketching a single *space-time curve*.

To illustrate my approach, consider the simple example of animating a flying dragon (Fig. 5-1). Animating the dragon requires the coordination of its shape over time as to follow the path's shape. With my approach, the basic animation can be created with a single sketched stroke. The stroke is used not only to provide the path of travel, but also to define how an abstraction of the character's shape (its line of action) changes over time. Additional strokes can be used to refine the movement, or add details such as the flapping of the wings. Creating such motion with existing techniques would require coordinating a large number of keyframes that specify deformations and positions along the path, or a method for puppeteering the degrees of freedom of the dragon.

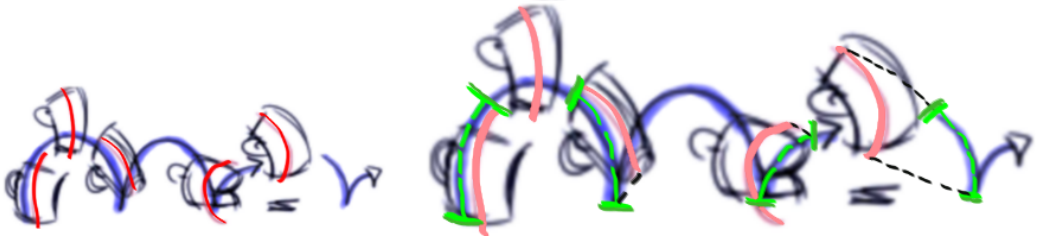

Figure 5-2: Inspiration: a hopping mug. The artist made the lines of action "dynamic" by having them match the blue trajectory (green marks were added on the right to show the matching parts). This shows that the blue stroke carries both shape and path information.

The key to my approach is an interpretation of the space-time curve defining a 2D dynamic skeletal line abstraction, or *dynamic line of action* (DLOA). From the STC, we extract the DLOA's *shape* at *discrete moments in time*, or *continuously over time*—depending on the curve's features. In simple cases, such as Fig. 5-1, a path-following DLOA is defined as a moving window within the parametric space of the stroke. The speed at which the stroke is drawn directly controls squash and stretch deformations. Looking closely at existing ways of sketching animations (see Fig. 5-2 and Fig. 5-6), we observe that the stroke encodes *shape* near singular points (Fig. 5-2) as well as between self-intersections (Fig. 5-6). We use these observations to directly define bouncing and rolling motions from the STC. The animation can be further refined by adding wave and twist specifications, by over-sketching lines of action or by adding secondary lines—possibly drawn from other viewpoints.

In addition to synthesizing a complete DLOA from a space-time curve, a second challenge is to animate a 3D character matching the shape of the DLOA over time. A frame-

by-frame computation of the character's 3D pose from its LOA using the matching method from Chapter 3 (Section 3) is not sufficient to produce smooth motions in the dynamic case. Minimizing for smoothness and using lower-dimensional splines helps ensure smoothness but makes it hard to get a tight match between the skeleton and the moving line, as done in the previous chapter (Chap. 4). Additionally, in previous chapters (3, and 4), I solved only for rigid bone transformations which does not allow squash and stretch. To solve these problems, I provide a robust line matching algorithm combining closed-form solutions with dynamic programming yielding a tight match between the character and the DLOA over time, possible including squash and stretch of the character.

I evaluated this approach with an informal pilot user study showing that space-time sketching is much faster than using keyframes alone—even compared with sketching expressive lines of action. Moreover, it shows that free-form animation is possible for beginners.

Technical overview

The main concept in this chapter is space-time sketching through a single stroke, called the *space-time curve*. From this curve and the speed at which it was drawn, we automatically initialize a *dynamic line of action* (DLOA), which itself drives a specific *body line* in the character model (e.g. spine, tail, wings, etc—or combination of these). We enable refinement of the resulting coarse motion through over-sketching the DLOA at specific times or adding secondary lines or controls.

Both the space-time curve and the DLOA are planar, defining projective constraints for the 3D animated character. This allows simpler user input and enables decoupling expressive motion specification—done from a given camera viewpoint—from a specific 3D character model. The same DLOA constraint can be applied to other body parts or to different characters, enabling the user to draft animations even before the character is fully modeled.

The different ways a dynamic line of action can be initialized from strokes are discussed in Section 2. In Section 2.1, I lay some foundations were strokes are used to control only shape. Then I build on this foundation to control both shape and trajectory with a single stroke (the *space-time curve* (STC) described in Section 2.2). The STC can be used to produce path-following DLOAs, critical to creating styles of animations such as Fig. 5-

1, where standard keyframing falls short. In the abstract form, the STC initializes DLOAs such as bouncing and rolling. I describe in Section 3 how to refine the animation by mixing and composing the resulting DLOA with other strokes, as well as controlling their twisting orientation via space-time cans. Lastly, in Section 4 I present a robust method to match a 3D character's skeleton to the DLOA.

2 Sketching dynamic lines of action

A dynamic line of action (DLOA) is a 2D **parametric surface** $x_{dloa}(s,t)$, where t is time and s is the parameter along the line. This surface can be displayed as a moving line by successively showing temporal iso-lines $x_{dloa}(s,t_i), i = 0...I$. Fig. 5-3 shows a visualization of the whole DLOA surface.

In this section, I describe how such a surface can be created from sketches. I first investigate the extension of the standard keyframing concept to LOAs (similar to last chapter). I then introduce a new abstraction, the space-time curve, and explain how it can be used to quickly draft and then refine the DLOA in a principled way.

The 2D stroke the user draws is denoted $C = [c_0, ..., c_{N-1}] \in \mathbb{R}^{2\times N}$ ($c_j \in \mathbb{R}^2 \,\forall\, j$), where the c_j are the samples recorded by the input pen device. A smooth curve is built with a set of piecewise linear basis functions $\{b_j\}_{j=0}^{N-1} \in [0,\ 1]$ centered at the parameterization points $U = [u_0, ..., u_{N-1}] \in [0,1]^N$, resulting in:

$$c(u) = \sum_{j=0}^{N-1} c_j b_j(u),$$

which can be written in matrix form, $c(u) = CB(u)$, where $B(u) \in \mathbb{R}^{N\times 1}$. By default, the parameterization is evenly spaced, i.e. $u_j = \frac{j}{N-1}$. As the user does not sketch at constant speed, the spacing between uniformly sampled points will vary, i.e. the curve will be locally squashing and stretching. Using a constant arc-length re-parametrization will produce uniform local lengths (eventually leading to uniform stretch of the character).

2.1 Keyframing lines of action

While this paper is focused on specifying DLOAs from a single space-time curve, I first describe the interpolation of static LOAs at key moments k, $x_{dloa}(s,t_k), k = 0...M-1$. The

user can create a DLOA by sketching multiple key-LOAs, as shown in Fig. 5-3.

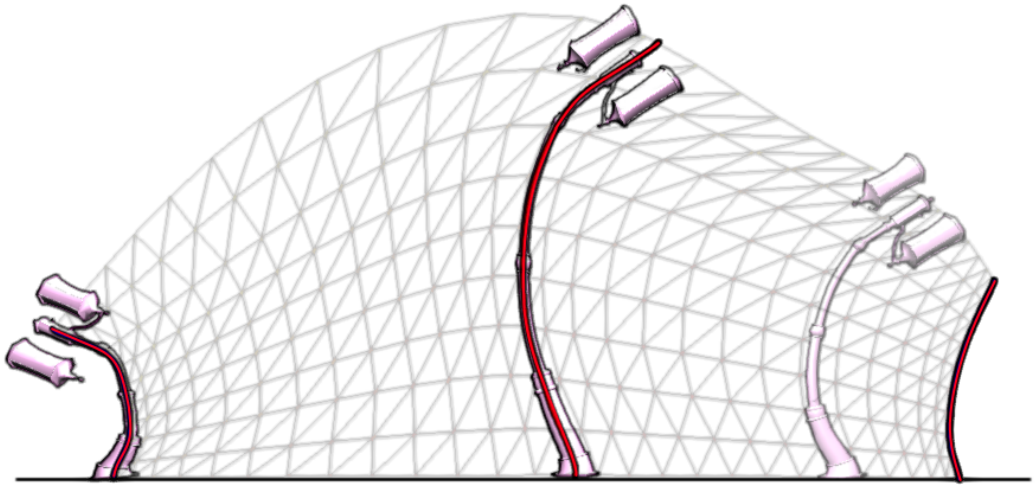

Figure 5-3: The user can create the space-time surface (DLOA) by sketching strokes (red) at key moments. Our matching method (Section 4) allows squash and stretch of the character's bones. Note that squash and stretch may require volume preservation which is not provided by linear blend skinning used in these figures.

Enabling this keyframing process to take place *in screen space* is different from using each key-LOA to pose the 3D character and interpolating the character's configuration space, as done in Chapter 3. 2D interpolation is an improvement over the prior approach as it can be used to ensure fluidity in the perceived motion, and has the advantage of decoupling animation specification from the 3D character model. However, two issues need to be solved for good quality interpolation between LOAs: length preservation and C1-smooth motion.

Local length-preserving interpolation. Static LOAs are 1D shapes. Interpolating two of them—best in a rigid as possible manner—requires *local* length preservation: if parts of the two extreme LOAs have the same length, the same part in all the intermediates should have the same length as well. Simply interpolating between their cartesian points (e.g. $x_{dloa}(s,t_k)$ and $x_{dloa}(s,t_{k+1})$) does not preserve length.

We can achieve length preservation by decomposing each LOA $x_{dloa}(s,t_k)$ into polar components as in [Sederberg and Greenwood, 1992, Alexa et al., 2000]—angles and lengths of each polyline—perform the interpolation, and then recover the interpolated curve by integrating the rotated polylines. The angles and lengths are respectively denoted $\Theta = [\theta_0, ..., \theta_{N-2}]$ and $L = [l_0, ..., l_{N-2}]$. Given angles lengths, and a root position

x_0, the curve is recovered with:

$$x_j = \sum_{n=0}^{j} l_n R(\theta_n)\, \bar{x} \; + x_0, \tag{5.1}$$

where $\bar{x} = (1, 0)$ is the reference x-axis and R is a rotation the 2D plane.

Then we compute the *absolute angle* between each polyline $x_{j+1} - x_j$ and the reference x-axis $\bar{x}$:

$$\{\theta_j,\, l_j\} \quad = \{\angle\,(\bar{x}, x_{j+1} - x_j)\,, \quad \|x_{j+1} - x_j\|\}\,. \tag{5.2}$$

C1-smooth motion. Linear interpolation of sparse keyframes can produce discontinuities at keyframe transitions—imagine a robot performing one action at a time. To allow creating fluent motion with few keyframes, we need an interpolation that provides smooth first derivatives (tangents). To do so, we use cubic Hermite spline interpolation, where the tangents at the key frames are computed automatically using Catmul-Rom averaging.

Suppose we want to interpolate between two key-lines $x(s, t_k)$ and $x(s, t_{k+1})$ over a time interval $[t_k, t_{k+1}]$. The polar components $[\theta(s, t_k)\; l(s, t_k)\, x(0, t_k)]^T$ are sequenced into a matrix $\Psi = [\Theta(s)\; l(s)\, X_0]^T \in \mathbb{R}^{3\times M}$ of M lines (in our example $M = 2$). We build a C1-smooth interpolation by sequencing the first derivatives (the tangents) $\dot{\Psi}$ to the right of the matrix $\left(\left[\Psi\, \dot{\Psi}\right] \in \mathbb{R}^{3\times 2M}\right)$, and using Hermite basis functions $B(t) \in \mathbb{R}^{2M\times 1}$:

$$\begin{bmatrix} \theta(s,t) \\ l(s,t) \\ x(0,t) \end{bmatrix} = \begin{bmatrix} \Theta(s) & \dot{\Theta}(s) \\ L(s) & \dot{L}(s) \\ X_0 & \dot{X}_0 \end{bmatrix} B(t). \tag{5.3}$$

Given the smooth angles, lengths and root positions $[\theta(s,t),\, l(s,t),\, x(0,t)]^T$, we compute a smooth dynamic line of action by following Eq (5.1):

$$x_{dloa}(s,t) = \int_{s=0}^{1} l(s,t) R(\theta(s,t))\, \bar{x}\, ds \; + x(0,t).$$

2.2 Space-time curves

Although keyframing is a useful building block for defining DLOAs, it cannot easily handle cases shown in Fig. 5-1 and Fig. 5-2, where shape and movement need to be carefully coordinated. The key idea to my approach is to enable the initialization of a complete spatio-temporal surface $x_{dloa}(s,t)$ from a single sketched stroke by re-interpreting the stroke parameter u into separate space and time parameters s and t.

We start by describing a simple path-following behavior, which is critical in generating styles of movement such as the one in Fig. 5-1. Then I extend the concept to non-degenerate initialization, producing bouncing (hopping) and rolling motions. In all cases, the resulting animation can be immediately shown to the user, and then refined by adding more strokes and controls (Section 3).

Figure 5-4: Left image: character model and space-time curve (STC). Right image: we dynamically warp (red window) within the parametric space of the STC to produce a dynamic line of action. Squash and stretch is controlled directly by drawing faster and slower.

Path-following. Consider Fig. 5-4 (top), where a long curve $c(u)$ is drawn, and the character (or line of action) is expected to move *through* the curve. This is achieved by defining a *moving window* within the parametric space of c:

$$x_{dloa}(s,t) = c(w(s,t)), \tag{5.4}$$

where $w(s,t)$ is a warping function that scales and shifts in time a parameter s. To compute $w(s,t)$, we first scale s with a constant b. To some extent, b reflects the character's length in parametric space (its computation being detailed bellow). Then we shift the window forward in time with t; only t has to be scaled, as to ensure the window remains within the STC's parametric space (e.g. $\subseteq [0,1]$)—leading to the following warping func-

tion, to be used as the parameter in Equation (5.4):

$$w(s,t) = t(1 - b\,s) + b\,s, \tag{5.5}$$

The scale b sets the length of the line $x_{dloa}(s,t_i)$ at a given time step t_i, and ultimately, of the character in screen space.

The choice of parameterization U for the input stroke also has an impact on the local lengths of the dynamic lines: as the user rarely draws at a constant speed (sample points along the curve are not evenly spaced), using a uniform parameterization U leads to local variations in length—useful for controlling squash and stretch along the dynamic lines. This can be removed if desired by re-parameterizing the curve U at constant arc-lengths.

A simple way to estimate b given an arbitrary parameterization U is to define it as the average length ratio between the skeleton's initial rest pose and the full length of the path:

$$b = \frac{\int_{s=0}^{1} \| \frac{\partial P_{vp} x_{body}}{\partial s}(s) \| ds}{\int_{u=0}^{1} \| \frac{\partial c}{\partial u}(u) \| du}. \tag{5.6}$$

Remembering that the DLOA $x_{dloa}(s,t)$ is a 2D *surface*, we can note that the simple initialization just described is very specific: it fits the full surface into a curve. I now describe two, more abstract ways of initializing DLOAs from a curve inspired by the way people tend to sketch motion (Fig. 5-2 and Fig. 5-7). In these sketches, the singular points and the self-interactions along the curve are interpreted as bouncing and rolling indications, used to automatically create a non-degenerated space-time surface for the DLOA. Note that in [Thorne et al., 2004] sketches are also used to specify motions, but only to select pre-defined motion clips. In this work, we formally establish a geometric link between the strokes and the shape of an abstract line; which in turn can be applied to arbitrary character morphologies.

Bouncing Let's consider the sketch in Fig. 5-2 used to represent a bouncing motion. In this case, semi-circles are drawn to specify contact intervals, interleaved by in-air paths. Important to note is how the semi-circles provide not only a rough trajectory, but also the *shapes* at takeoff and landing (red lines of action).

Inspired from this sketch, I provide an automatic way to synthesize a DLOA from a space-time curve $c(u)$ that holds singular points. From this single stroke, we extract both

shape (the red lines) and trajectory, as well as timing—which is determined from the time spent at the contact points when drawing the stroke.

I detect singular points along the curve and define a suitable warping window (Equation (5.5)) to pick-out individual key LOAs (the red curves touching the singular points in Fig. 5-2), while setting their respective timing via the speed at which the curve was drawn. Interpolating these key LOAs using Equation (5.3) gives a first *intermediate* DLOA that is denoted $\hat{x}_{dloa}$. It has the right shape over time, but not the right trajectory yet—at least not in the air stage between contact points.

We now adjust the trajectory of the intermediate DLOA $\hat{x}_{dloa}$ as to approximately match the trajectory conveyed by the stroke. The solution is based on two steps: first, we detect which point $s^*(t)$ on the line needs to follow the trajectory, based on the off-ground angle (detailed below). The second step is to ensure smoothness between contact and flight stages by constructing a trajectory $T(t)$ that will smoothly link 3 points for each semicircle: takeoff $c(u_1)$, middle of semicircle $c(u_2)$, and landing $u_{air} = (u_1 + u_2)/2$.

Given the point $s^*(t)$ in the dynamic LOA that is to match the trajectory $T(t)$, we define the constraint $T(t) = x_{dloa}(s^*(t), t)$. The constraint is linear and can be met by adding a correction term $\Delta x(t) = T(t) - \hat{x}_{dloa}(s^*(t), t)$, to the intermediate $\hat{x}_{dloa}(s^*(t), t)$; resulting in the final bouncing DLOA:

$$x_{dloa}(s, t) = \hat{x}_{dloa}(s, t) + \Delta x(t). \tag{5.7}$$

The point in the line (eventually in the character's body) $s^*(t)$ that follows the trajectory is computed based on the off-ground angle of the semicircle. For angles smaller than a threshold, $s^*(t)$ is the bottom tip of the LOA, and it goes to the middle of the line for a vertical takeoff. The coordinate $s^*(t)$ interpolates between the coordinate at the takeoff keyframe s_1^* and the coordinate at the landing keyframe s_2^*, resulting in: $s^*(t) = s_1^* t + (1 - t)s_2^*$.

Rolling Another abstraction is when the user sketches a loop to convey a rolling movement (see Fig. 5-7). Indeed the loop could be interpreted as a single rigid rotation. However, looking closely, we see how the upper body of the character matches the *shape* of the loop over time (dashed line in the right of Fig. 5-7). This is similar to the *path-following* behavior described earlier, only the difference is that in the rolling case, the

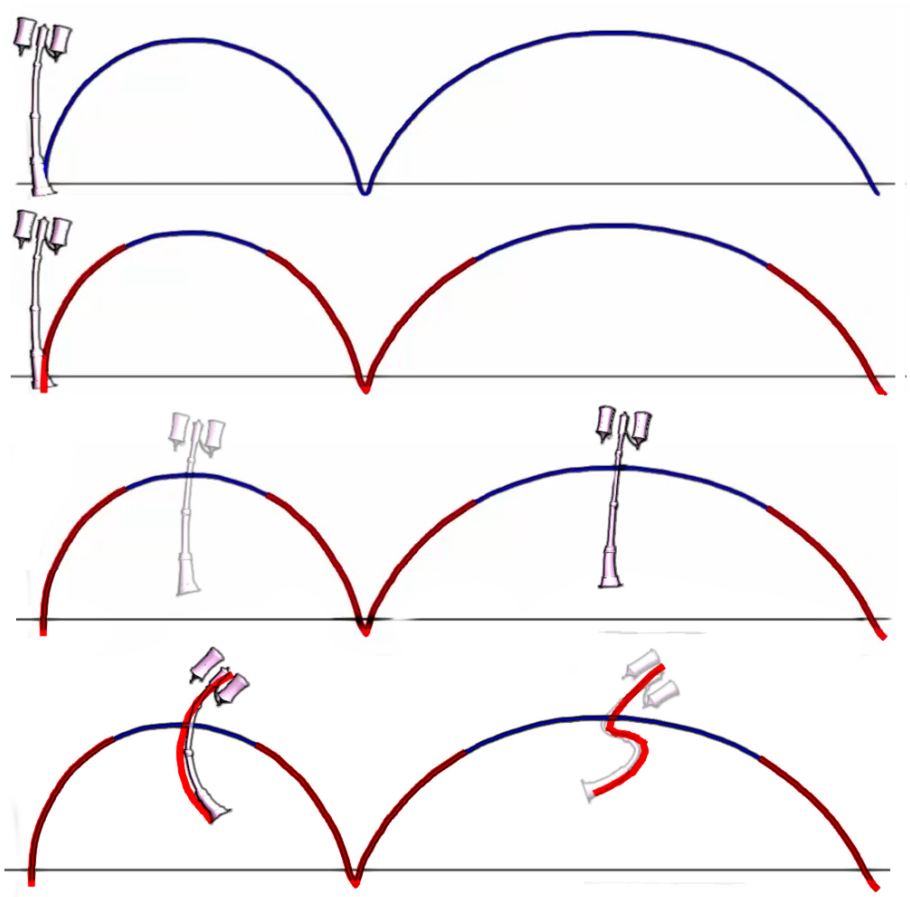

Figure 5-5: Our method parses the space-time curve for singular points, and uses a warping step function to pick individual key frames (red lines)—which are then interpolated to provide the shape of the character over time, while the trajectory is determined by the path. First row: the user sketches a space-time curve. Second row: the key frames that are automatically picked out. Third row: interpolated motion with a trajectory correction. Fourth row: the user edits the initial motion by over-sketching keyframes on the space-time curve—automatically providing its timing.

trajectory and shape do not coincide; if a point in the body did follow the red curve in Fig. 5-7, the body would move backward to the left before continuing its flight to the right. In other words, between self-intersections we have a shape-following constraint and the sketch only roughly indicates a trajectory. I use this interpretation of rolling abstractions to automatically extract a rolling DLOA from a space-time curve $c(u)$ holding self-intersections as follows.

To fulfill the dynamic shape-matching constraint between intersecting points, while blending with the shapes at the landing and takeoff (similarly to the bounce near singular points), we blend between a path-following DLOA $\bar{x}_{dloa}$, defined over the looping region,

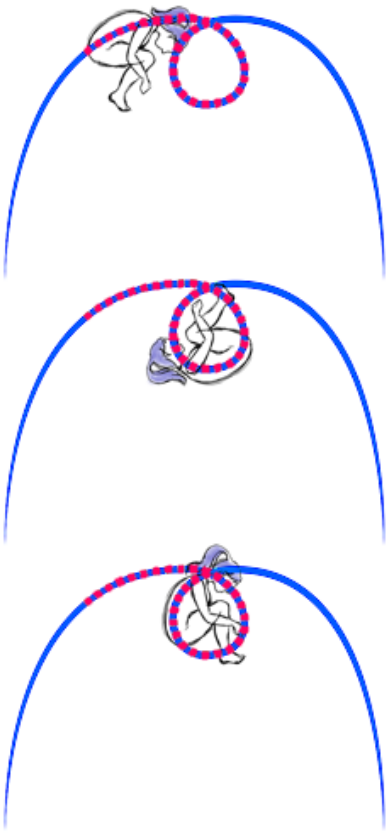

*Figure 5-6: In these artist sketches, the character's upper body matches the **shape** of the red curve between the two self-intersecting points (area outlined by the dashes). Supposing the character's trajectory was matching the path, the body would be globally moving backward (see third row compared to second and first), while the true global trajectory is ballistic on the horizontal axis and only goes forward. In Fig. 5-7 we can see the character's global trajectory in green, which does not strictly match the drawn path.*

and a bouncing DLOA $\check{x}_{dloa}$, defined from the takeoff and landing keyframes, resulting in an intermediate DLOA $\hat{x}_{dloa}$—that we finally correct for trajectory (to follow the green trajectory in Fig. 5-7, using eq.(5.7).

Figure 5-7: Shape and trajectory are both encoded in the roll abstraction. In Fig. 5-6, we can see how the shape of the character dynamically varies to match the shape of curve. In this drawing, we can see in green the trajectory the character should follow and how it only approximates the overall path drawn by the artist.

First, we detect singular points (indicating takeoff and landing) and self-intersections in-between. Let (u_0, u_1, u_2, u_3) be the four associated parameter values, where $c(u_0)$ is the takeoff point, $c(u_1) = c(u_2)$ is the self intersection point, and $c(u_3)$ is the landing point.

The path-following DLOA $\bar{x}_{dloa}$ is extracted using a continuous warping (eq. (5.4)) within the looping region of $c(u)$, i.e. between u_1, and u_2. The bouncing DLOA $\check{x}_{dloa}$ is defined as the interpolated keyframes extracted at takeoff and landing coordinates u_0 and u_3. With these two different DLOA motions, we perform a blending in order to go from the takeoff shape, to the rolling motion, and back to the landing shape, following a classic blending operation:

$$\hat{x}_{dloa}(s,t) = t\ \check{x}_{dloa}(s,t) + (1-t)\ \bar{x}_{dloa}(s,t), \tag{5.8}$$

where t is equal to 1 at takeoff (influence interval for the first keyframe), then smoothly drops to zero, which sets the DLOA to $\bar{x}_{dloa}$ (path-following), before the influence of the second keyframe becomes non-zero and increases to 1. To perform this blending, we use our length-preserving interpolation equation (5.3), where t and $1-t$ in eq. (5.8) are replaced by their Hermite basis function analogs $B(t)$ in eq. (5.3).

Lastly, the resulting (intermediate) DLOA $\hat{x}_{dloa}$ needs to be corrected in order to prevent the backward motion we already mentioned. This is done by constraining the center of mass $\hat{m}(t)$ of $\hat{x}_{dloa}$ to follow the green trajectory in Fig. 5-7. We build the green trajectory $T(t)$ by interpolating the three points $c(u_0)$ (takeoff), $c(u_{air})$ and $c(u_3)$ (landing), with $u_{air} = (u_1 + u_2)/2$. We then add the correction term $\Delta x(t) = T(t) - \hat{m}(t)$ to $\hat{x}_{dloa}$, where $\hat{m}(t)$ is the intermediate center-of-mass after the blending operation eq.(5.8); resulting in both the expected trajectory *and shape*—over time—as shown in Fig. 5-8.

These constructions show that different DLOAs can be easily blended over time. Next, I re-use and extend this methodology in order to allow the user to refine and extend the rough animation he just created.

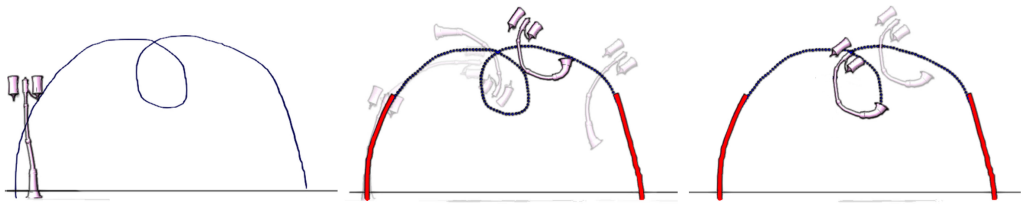

Figure 5-8: The user sketches a loop shape (first column on the left). The resulting motion blends between the key frames at the side (second column) and a path-following DLOA in the middle section of the stroke—between self-intersecting points. Third column is the intermediate DLOA before a trajectory correction.

3 Refining motion

3.1 Over-sketching keyframes

A natural way for a user to edit an animation is to stop it and over-sketch one or several keyframes. Implementing this is straightforward when the DLOA was created using the keyframing approach in Section 2.1: the user selects or inserts a keyframe at a time t_k, and over-sketches the current line of action $x_{dloa}(s,t_k)$ displayed by the system. Time intervals are automatically re-set when a new keyframe is inserted. The user can edit timing by sliding the t_k values along a timeline.

This type of interaction is also supported when the DLOA is created from a space-time curve (STC) (Section 2.2). A further benefit in this case is that the STC can be used as a visual timeline where keyframe strokes can be directly sketched-over. As each point u_i in the STC maps to a time value t_i, we can identify the closest point u_* to the drawn stroke, providing us with a time t_* where the new keyframe is to be inserted. Sketching over the space-time curve greatly eases shape and motion coordination. However, since the DLOA now holds continuous motion, blending is required to take the new keyframe(s) into account.

Blending DLOAs. Let $[t_{k-1},t_{k+1}]$ be the interval of influence of a new key-stroke (the user can edit this interval using the timeline). Let $x^1_{dloa}(s,t)$ be an existing DLOA on which is sketched a second stroke $x^2_{dloa}(s,t)$ defined as a constant keyframe shape. The goal is to transition from x^1_{dloa} to x^2_{dloa} over $[t_{k-1},t_k]$, before transitioning back to x^1_{dloa} over $[t_k,t_{k+1}]$. This is done by using Equation (5.8), where $\bar{x}$ is replaced by x^1 and $\tilde{x}$ by x^2. A result is shown in Fig. 5-9, where the dragon's tail first follows a path, is then constrained by the

sketched curve at t_k, and then goes back to the path-following.

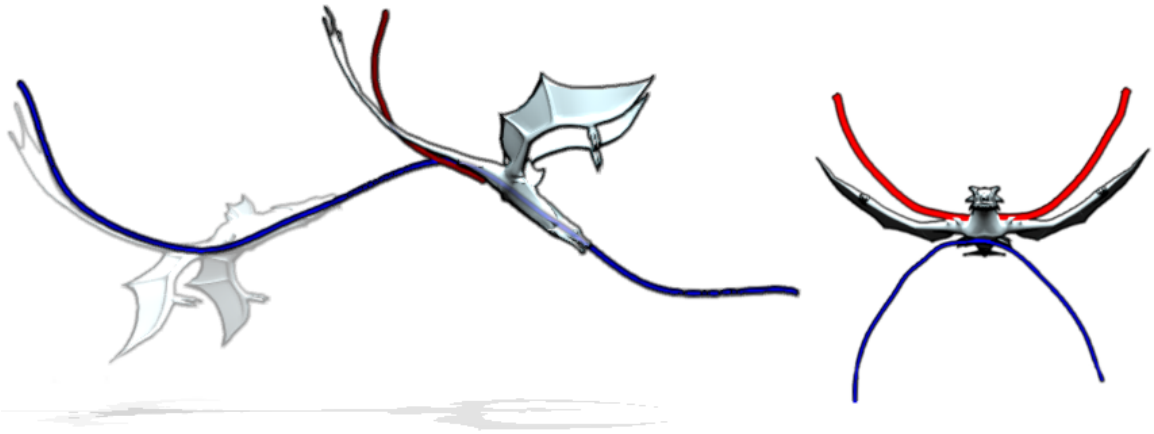

Figure 5-9: The user sketches a line of action stroke on top of a path-following DLOA to alter the motion of the tail over a time interval. The path-following motion (a DLOA) blends with another DLOA, the static key frame sketched for the tail, over a time interval. Right: the user edits secondary lines onto a separate plane and view.

Sketching secondary lines. Another type of refinement consists in animating secondary lines of action by sketching keyframes from other viewpoints. This can be used to animate secondary body-lines such as the wings of a dragon. To do this, the user chooses another viewpoint, plays the animation, stops it when desired and over-sketches a few key-LOAs for the secondary line. The secondary keyframes are thus easily synchronized with the main DLOA. They are interpolated using the method in Section 2.1 and can be played in loop if desired. See the image on the right in Fig. 5-9.

Note that the over-sketching method replaces the motion in an absolute way, i.e. if we re-draw the STC, the tail will remain oriented the same way. Another method is to treat the over-sketching as a displacement map layered on top of the coarse STC, which is explored with wave refinements in the next section.

3.2 Wave curves

A *wave curve* is a layered refinement enabling the animator to edit the current DLOA in a relative manner. It allows adding periodic wave motion to the DLOA, by composing it with a periodic displacement map. To illustrate this concept, let us consider the following scenario: instead of sketching a path and having a line of action follow it as in the path-

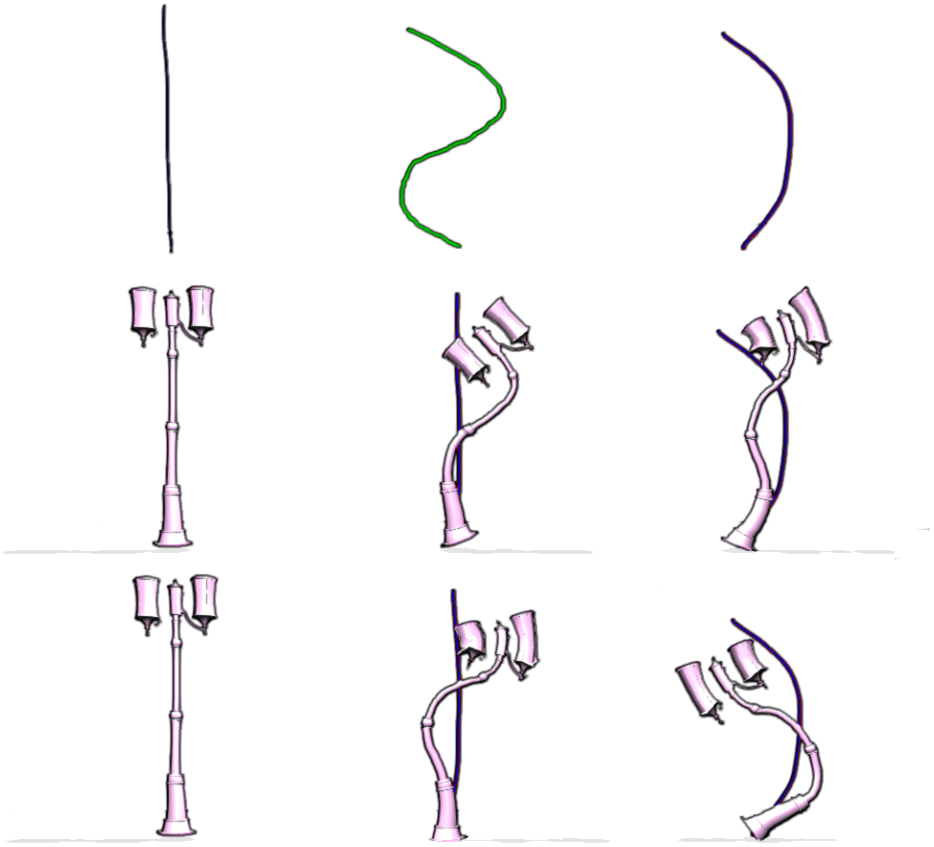

Figure 5-10: First column: a straight static line. Second column: a periodic wave curve (green) has its transformation (local angles) defined relative to the straight line. Third column: the relative transformation is transferred to another (bent) line by adding its angle offsets to the bent line.

following behavior (Section 2.2), we draw a curve and make the curve *move through* the line of action over time, as shown in the first column of Fig. 5-10. This type of behavior is often seen in spine-driven locomotion where a tail or flag follows an oscillating driving mechanism.

Combining the current DLOA and this wave motion by directly applying the previous blending method would completely replace the DLOA since the wave motion is periodic and covers the full time range. In contrast, we would like to combine them in a way that preserves the existing DLOA and allows subsequent editing of both the existing coarse DLOA (e.g. by re-drawing keyframes) and the wave refinement (see Fig. 5-10).

To enable this, we define the motion of the wave (i.e. its shape over time defined by angles $\theta(s,t)$) as relative—or as an offset $\Delta\theta(s,t)$—to a reference curve x_{ref}. This ref-

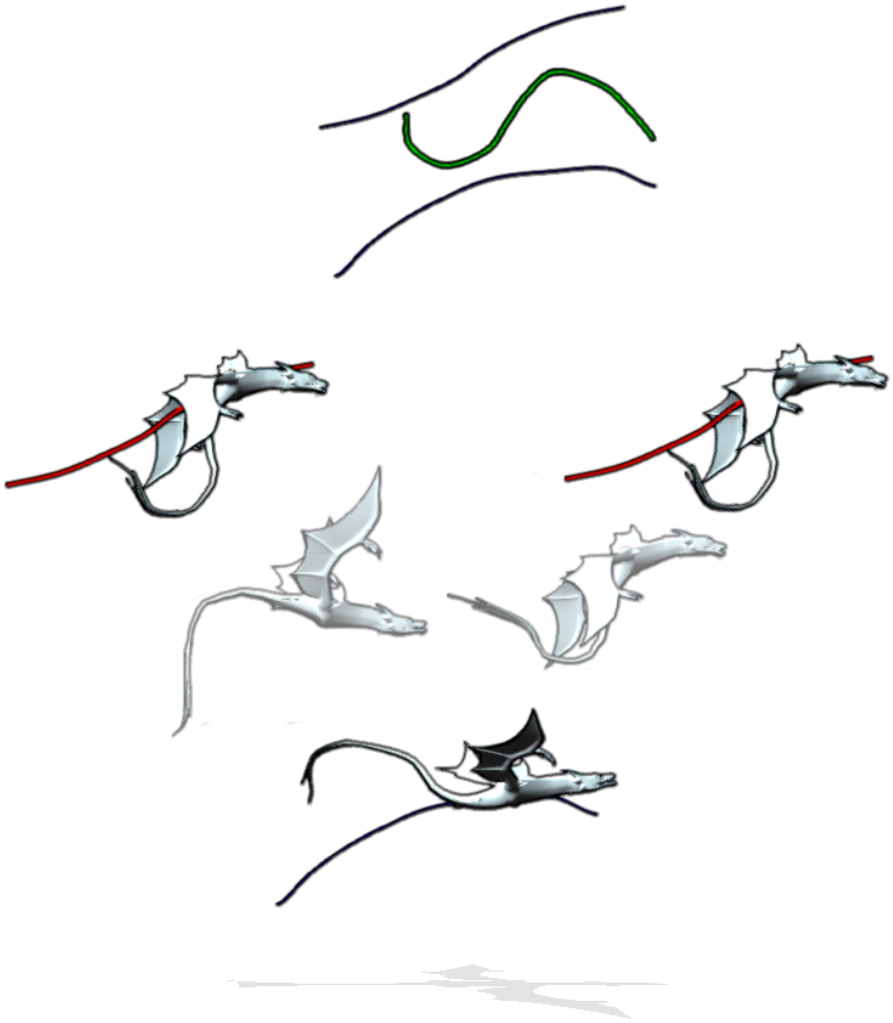

Figure 5-11: Composing two layers of motion can facilitate creating complex motions such as a flying locomotion. This examples was created with a periodic wave layer on top of a keyframed layer defined by two strokes.

erence shape should be coarser and representative of the more complex wave motion. As a reference shape, I use the vector defined between the two end-points of the wave stroke $x_{ref} = c(1) - c(0)$, as it naturally provides a coarse representation of the wave. We compute the relative angle w.r.t. the reference axis as follows:

$$\Delta\theta(s,t) = \angle\left(x_{ref}, \frac{\partial c(w_\pi(s,t))}{\partial s}\right), \tag{5.9}$$

where w_π is a periodic warping function that evolves over time. Finally, the angle

relative to the reference axis $\Delta\theta(s,t)$ is composed on top of an animated DLOA by *adding* the angle to the shape blending Eq. (5.3), which is done by setting its temporal basis function to $b_{wave}(t) = 1$. Fig. 5-10 depicts wave curves composed with keyframed DLOAs.

3.3 Twisting around a space-time curve

I use the "two cans" drawing tool described in the previous chapter (Section 3.1) to allow the user to specify twist along the space-time curves (Section 2.2, as shown in Fig. 5-12). The cans control a single angle γ_i that rotates around the stroke. The user adds cans along the strokes, which are then interpolated into a smooth twist function $\gamma(s)$. Different key-LOA strokes over time define different $\gamma_j(s)$ to be interpolated, while adding cans to a STC directly defines $\gamma_j(s,t)$. I blend between all the active cans at a time t using the same Hermite based interpolation as in Equation (5.3). The smooth twist angle $\gamma(s,t)$ is then used along the dynamic line of action $x_{dloa}(s,t)$ to control the character's skeleton.

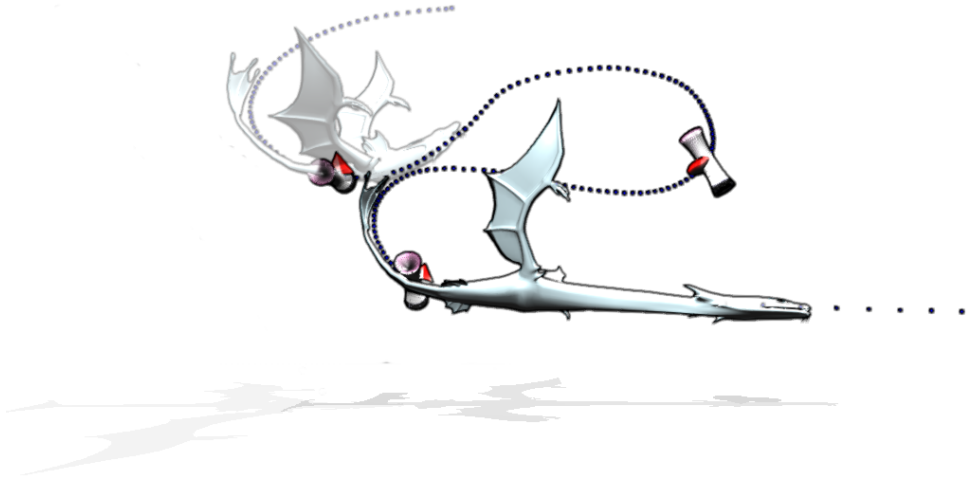

Figure 5-12: The cans are specified at space-time coordinates along the space-time curves. In the bottom-right, we can see non-uniform stretching of the character as it gradually enters an area of the curve that was drawn faster.

4 Dynamic line matching

The 2D dynamic line (of action) $x_{dloa}(s,t)$ corresponds to a *projected* skeletal *line* abstraction for parts of the character's body such as: the spine, tail, wings, arms, or combinations of them—called the body line in Chapter 3.

Unfortunately applying the method in Section 3 leads to inconsistencies between

frames. Two aspects of the algorithm cause these the solver to fight between solutions depending as the shape of the DLOA evolves over time. The first one is the fact that it solves for the orientation only which makes it hard to exactly match the LOA at a given instant in time. The second aspect is the fact that it minimizes the average distance between lines while trying to ensure that shape matches with a soft penalty—causing the solver to fight between constraints.

Instead of turning the hard constraint into a penalty, I solve for the lengths of the bones and orientations using closed-form solutions, combined with dynamic programming where the optimal solution for the skeleton is the set of optimal solutions starting from the root of the chains. I also remove the average distance and replace it with a single point touching the line. I will use the root of the body part as the point in the skeleton that has to touch the DLOA.

Similar solutions exist in the computer vision community, where dynamic programming offers a global solution to the matching of 2D pictorial elements [Felzenszwalb and Huttenlocher, 2005]. My solution is based on the idea that we can project the bones onto a 2D plane and solve the 2D problem using closed-form solutions, and then transfer the quantities back into the 3D pose, and then proceed to the next bone down the chain.

Hence I use the viewing plane constraint introduced in the previous Chapter 3, Section 2, resulting in the following dynamic line matching problem:

$$\min_{w_u(t),w_v(t),\mathbf{\Theta}(s,t),\mathbf{L}(s,t)} \quad \int_t \|\mathbf{Px}(s^*,t) - \mathbf{x}_{dloa}(s^*,t)\|^2 \; dt,$$

$$\text{subject to} \qquad \frac{\partial \mathbf{Px}}{\partial s}(s,t) = \frac{\partial \mathbf{x}_{dloa}}{\partial s}(s,t), \; \forall s,t.$$

where we need to solve for the root trajectory $w_u(t)$ and $w_v(t)$, joint angles $\mathbf{\Theta}(s,t)$ and local lengths $\mathbf{L}(s,t)$ along the line. The position s^* that has to touch the line is the root of the body part.

In the next Section (4.1), I describe how to solve exactly for relative transformations of each bone constrained by the DLOA at time t. This exact algorithm rapidly and robustly solves for the skeleton matching the DLOA while allowing squash and stretch by adjusting the length of the bones. Fig. 5-15 shows the result when solving for relative angles of the joints in the case of a humanoid. Note however that my results were produced

with linear blend skinning of the surface geometry which does not preserve volume—a feature essential for cartoon squash and stretch effects.

4.1 Exact solution for relative transformations

When each bone rotation is a relative to a parent orientation, we can start from the root of the kinematic tree, solve for the relative angle and bone length, transfer the quantities into 3D, and proceed to the next bone down the chain up to the leaf of each chain. The process starts by setting the root position to its corresponding position on the line.

We denote the lengths of the bones as $l_i(t)$, the relative orientations $q_i(t)$, and the absolute orientation $Q_i(t)$. With the viewing plane constraint, we are looking for a single (angle $\theta_i(t)$) rotation in the viewing direction v_{dir}, i.e. $Q(\theta_i(t), v_{dir})$, that rotates the bone parallel to the viewing plane to match its corresponding line direction (see Fig. 5-14).

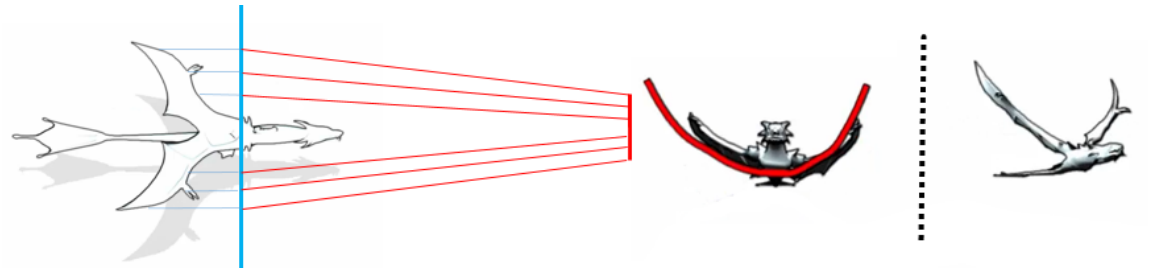

Figure 5-13: Both the body line and the DLOA at time t are projected onto a plane located at the root of the body part. For sketching secondary lines such as the wings, the character's main line (e.g. the spine) remains in a fixed reference pose. Last image on the right is the pose at the time of the sketched DLOA key frame.

The first step is to project both lines onto a single plane. We start by inverse projecting the DLOA (red stroke in Fig. 5-13) onto a plane located at the root of the body part and oriented in the viewing direction. We cast rays along the DLOA $x_{dloa}(s_i, t)$ to recover the corresponding positions on the plane denoted $z_i(t)$. We then project the bone positions x_i onto the same plane using the projection operator denoted P (blue in Fig. 5-13).

For each bone i, we compute the angle $\theta_i(t)$, and then recover the bone's relative orientation w.r.t. to the parent orientation $Q_{i-1}(t)$:

$$\begin{aligned} \theta_i(t) &= \angle(Px_{i+1}(t) - Px_i(t), z_{i+1}(t) - z_i(t)), \\ q_i(t) &= Q_{i-1}^{-1}(t) Q(\theta_i(t), v_{dir}) Q_i(t). \end{aligned}$$

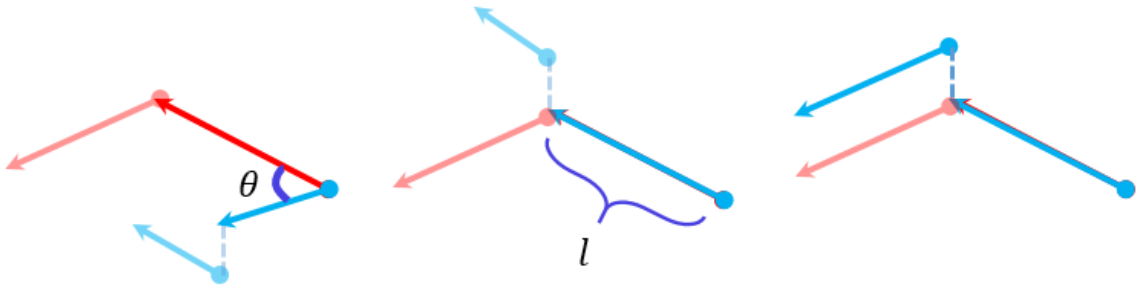

Figure 5-14: Our skeletal line matching: blue and red are the projected bones and DLOA directions. Dashed lines are offsets in the character's kinematic tree. For each bone starting from the root, we compute the angle, then adjust its length, and we repeat for each bone down the chains.

For the length, we seek to preserve the length ratio between the initially projected skeleton bone direction, and the length of the non-projected bone direction. Given the initial *projected* bone direction length $a_i(0) = \|Px_{i+1}(0) - Px_i(0)\|$, the initial bone direction length $l_i(0) = \|x_{i+1}(0) - x_i(0)\|$, and the sketched bone direction length at time t, $a_i(t) = \|z_{i+1}(t) - z_i(t)\|$, we have the relation:

$$\frac{a_i(0)}{l_i(0)} = \frac{a_i(t)}{l_i(t)}, \tag{5.10}$$

and thus by extracting $l_i(t)$, we obtain:

$$l_i(t) = \frac{\|Px_{i+1}(t) - Px_i(t)\|}{\|z_{i+1}(0) - z_i(0)\|\|x_{i+1}(0) - x_i(0)\|}. \tag{5.11}$$

To prevent drift from accumulating, we initialize the orientations and lengths at each frame with an initial pose, i.e. $q_i(t) = q_i(0)$, and $l_i(t) = l_i(0)$.

Including the twist: We incorporate the smooth twist angle $\gamma(s,t)$ from Section 3 by computing the differential twist angle of each segment $\Delta\gamma_i(t) = \gamma(s_i,t) - \gamma(s_{i-1},t)$, and applying a local rotation around the local bone direction $\bar{x}$:

$$\hat{q}_i(t) = q_i(t)Q(\Delta\gamma_i(t), \bar{x}). \tag{5.12}$$

Secondary parts are automatically taken into account due to the hierarchical structure of the skeleton. For instance the wings are attached to the spine of the dragon in Fig. 5-9, and inherit the parent's (spine) orientation, including the twist. Note, the length of

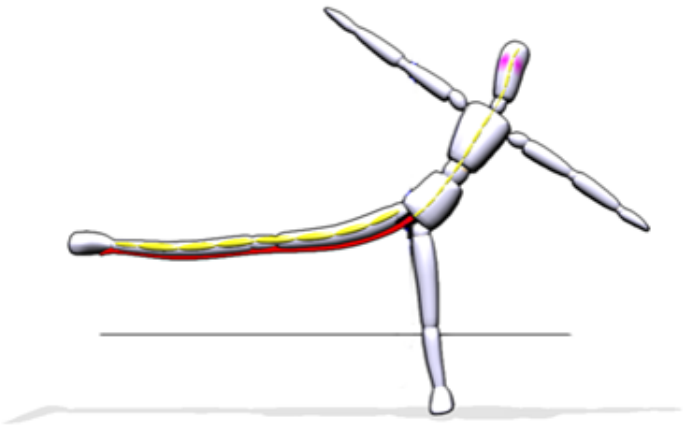

Figure 5-15: Solving for relative orientations (of each joint), the matching method can effectively solve for skeletons that have offsets in the kinematic tree, such as between the pelvis and upper legs in humanoids. The method also allows stretching the bones robustly.

parent bones is not propagated to children bones with this formulation.

5 Results

Experimental results shown in the Chapter and the accompanying video were created using my prototype implementation of the space-time sketching tools described in Sections 1, 2 and 3. In this section, I briefly describe the prototype and its user interface, then describe a user study we conducted with both novice and expert animators to assess the usability of the space-time sketching tools.

5.1 System description

The user is presented with a canvas for drawing keyframes and space-time curves, and a timeline for changing their timing. By default the user draws keyframes. A special key is pressed to sketch a space-time curve. The system automatically classifies the space-time curve as bouncing, rolling or path-following based on the geometry. The user draws over the path to automatically insert a keyframe, whose timing is retrieved from the path.

The user is first presented with a character in a reference pose and starts sketching. When the user selects a body part by pressing a key and drawing over the reference pose, the system snaps a plane onto the root of the body part, and is ready to animate the character using our method in Section 4. At this point, the sketches are inverse-projected

onto the *action* plane, and linked to a body part.

Because the 2D sketch is decoupled from the 3D counter-part, the user can apply the same line motion to to a different character, given the same viewpoint. To edit secondary parts of the body, the user can add a second sketching panel, and rotate the camera to a side view. The secondary sketches are freely sketched and the user then selects the body part which will snap the plane onto the root of the body part. The system will automatically snap the secondary sketch to the position of the primary sketch. This is done by taking the point at the intersection of the secondary plane and the primary DLOA and assigning it to the root of the secondary line. In my system, the user can chose to see the constrained parent-child sketches or the free-floating sketch; which is the default behavior.

Many of the sketching tools described in this chapter evolved as a consequence of user interaction and feedback. For instance, I initially offered only a basic keyframing interface for creating animations such as the one in Fig. 5-1 and observed that users were attempting to draw paths. After I introduced the path-following STC, users started drawing abstractions such as bounce and roll doodles—expecting my system to understand. I asked people to draw over a dragon flying animation, resulting in the wave shape that I implement as a periodic path to create the animation in the video and Fig. 5-10.

5.2 Evaluation

As an initial evaluation of this approach, we ran an informal user study with eight participants with varying levels of expertise in animation, ranging from complete beginners to professionally trained animators. We asked each participant to reproduce three short animations, first by keyframing lines of action ; then again by sketching space-time curves. The target animations videos were not made with our system. Animation (1) is a path-following motion of a dragon with a lot of twisting. Animation (2) is a flying locomotion of the same dragon. Animation (3) is a lamp bouncing animation.

Participants had never seen or used my system. They were each given twenty minutes to learn the basic keyframing and space-time sketching tools in our system: the participant was presented with the basic operations of adding keyframes to a timeline; sketching the strokes; manipulating the twist cans; and editing secondary parts (e.g. the wings). Then we showed each target animation and asked the participant to reproduce

it, first with keyframing and then with space-time curves. In each case, the participants were allowed to add twist cans. After completing each task, participants were asked to self-evaluate the quality of their results on a scale from 0 to 5. Note that our experiment did not control for order effects. In each case, we computed the time spent to complete the task, the number of pen clicks, and the number of entities (strokes and cans) used to obtain the final animation. The results are as follows:

Animation	time	entities	clicks	score
dragon1	353 s.	13	95	1.6
dragon2	321 s.	9	69	2.1
lamp	260 s.	11	55	3.3

Table 5.1: User response to keyframes (K): time spent, number of keyframes and twist cans, number of clicks, and self-evaluation scores per participant.

Animation	time	entities	clicks	score
dragon1	139 s.	4	34	4.2
dragon2	229 s.	2	40	3.6
lamp	161 s.	5	48	4.5

Table 5.2: User response to space-time curves (STC): time spent, number of curves and twist cans, number of clicks, and self-evaluation scores per participant.

We can see that space-time curves help the users achieve results quicker than keyframes in all cases. We observed average speed-up factors as high as 20 between sketching keyframes and sketching path-following curves. The effect is less visible when reading the total time because participants can spend more time on refining and twisting the curves. Space-time curves also receive significantly higher self-evaluation scores for all three animation targets. While some of this may be explained by order effects, we feel the effects are strong enough to warrant further study.

We observed novice users tended to be discouraged quickly by keyframing. Most of the time, they were not able to finish the task. Some participants gave up after trying for five minutes or more. In contrast, the space-time curves allowed all participants to quickly obtain results that matched their desires and expectations, encouraging them to spend more time polishing their animations. The twisting tool was used consistently and comfortably by most participants. Some participants had difficulty using the wave curves and found them counter-intuitive, although this may be an issue with the system's interface and our training procedure.

Advanced users performed slightly better with the keyframing tools than novice users,

but still preferred to use the space-time curves, as they obtained their results quicker and more naturally. They also liked the fact that they had instantaneous results with space-time curves followed by refinement and twisting.

6 Limitations and future work

Using sketching to freely create and refine 3D animations is a difficult problem. In this chapter, we used sketching to specify 3D animations from a given viewpoint, therefore limiting ourselves to projective control. To abstract the character's body, I used the notion of a moving line of action allowing the same moving LOA to be applied to different body parts or characters.

Although possible, I showed that defining a DLOA only from keyframes (temporal slices) is not the best way for coordinating motions. Taking an analogy from geometric modeling, this method is similar to shape design through sketching slices: getting a smooth result is not easy, and the result can only be seen when all the slices have been defined. In contrast, this new space-time abstraction for sketching motion (the STC) enables an animator to initialize a full coordinated motion with a single stroke—providing immediate display and allowing for rapid refinement.

This method brings several limitations. I focused on the motion of a single line and illustrated the concept with simple character morphologies (lamp, cactus and dragon). We can use the body swapping method from the last chapter (see Chap. 4, Fig. 4-4) to semi-automatically animate a multi-legged figure (e.g. humanoid, or dog) from a single moving line. It would require more work to so automatically.

I defined three specific ways of creating a DLOA from a space-time curve: path-following and two richer mechanisms for extracting coordinated motion and trajectory from intuitive strokes. The resulting motion can then be refined, which increases the range of possible results. While we can detect which behavior to adopt between the bouncing, rolling and path-following based on the geometric features found in the strokes, I did provide an automatic way of between wave curves and other unit keyframes.

The focus of this work is on planar movement plus out-of-plane secondary motions and twist, since many motions meant to be seen by viewers are in fact planar. I think the dynamic LOA models could be directly extended to 3D curves; I see no reason why

not. However, the actual editing of 3D curves remains a challenge today. Another route to 3D motion could be to use our new 2D dynamic line constraints along data-driven motion priors in a space-time optimization framework [Min et al., 2009]. However, one of the main reasons we devise free-form tools for character animation is for the ability to produce arbitrary movements that may include squash and stretch effects.

7 Conclusion

I introduced the new concept of space-time sketching for free-form animation of 3D characters. Thanks to this new concept, a coarse but coordinated animation—possibly including squash and stretch—can be drafted with a single stroke. Other strokes and controls can then be added enabling the user to progressively sculpt and refine motion.

The animation is designed by sketching in 2D from viewpoints of interest: it defines a dynamic line of action used as a projective constraint that robustly drives the 3D character. The resulting independence between animation control and the 3D character model enables the user to simultaneously edit the 3D model during the design stage.

At last, a new motion sculpting metaphor is a promising way of improving the ease with which we can create quality animations.

6 | Conclusion

Computer animation holds tremendous potential for expressing and enhancing our imagination. However, the current free-form animation tools do not allow easily fulfilling high-level artistic principles that help creating quality poses and animations. This thesis adapts commonly drawn shape and motion abstractions for 3D character posing and animation while remaining flexible. In other words, the tools provided in this thesis allow stretching body parts and animating imaginary characters such as dragons. According to the user evaluations performed in Chapters 3 and 5, these new tools have the potential of helping experts and beginners alike achieving better quality animations quicker.

1 Contributions

The line of action allows an animator to create more readable and expressive poses by directly sketching its overall flow with a single stroke. Realizing this concept required formalizing the line of action concept and providing a mathematical definition relating the character's shape to the shape of LOA. The projective nature of the line of action can easily lead to data-driven solutions for the pose synthesis. I managed in this thesis to reduce the size of the problem while allowing flexibility in the resulting poses by constraining the solution to viewing plane deformations—making this LOA posing tool compatible with the free-form animation tool set.

I explored the idea of adding dynamics to the line of action strokes allowing the line to be unleashed as an elastic ribbon. In this same work, I paved the way for driving the motion of multi-legged characters with a single moving line. How to automatically change body parts dynamically and interact with the environment are interesting questions for future investigation.

Finally, the space-time curve allows an animator to draft a full coordinated motion with a single stroke. The power of the motion models for the dynamic line—used as a projective skeletal line constraint—is that they are independent from specific body parts or characters and can be used in a free-form animation setting. The drafted motion can then be progressively refined with extra strokes and twist cans. I showed that different line motions can be sketched and composed together leading to control over rich motions with few strokes.

2 Future Directions

Automatically beautifying shapes

If we can automatically compute curve features from 3D shapes, then we can use the line of action constraint from chapter 3 together with a metric of how close is a curve to a beautiful curve—namely how close it is to a family of C- and S-shaped curves—to automatically beautify the shape.

One possibility to compute curve features is to skeletonize the shape. However, the problem with medial skeletonisation techniques (e.g. using the L2 distance) is that the computed skeleton has too many branches and is typically noisy. To automatically compute a skeleton that could be driven by a line of action requires filtering and pruning the skeleton to remove the useless branches. One way to do this could be to look for a main line. Recent work [Huang et al., 2013] uses the L1 distance to compute a curve skeleton, and the work of [Zheng et al., 2015] automatically computes a "backbone" which could be naturally deformed by a line of action.

We could also imagine extending this idea to beautifying contours and flowing between multiple shapes and characters.

Camera placements

The lines of action constraint introduced in Chapter 3 can be used to produce a more aesthetic pose from a given viewpoint. This idea can be taken in the other direction: given a pose or motion, find the most aesthetic *viewpoints* by taking the pose that admits the LOA which is closest to the family of C- and S-shaped curves. This aesthetic principle could be combined with other textbook photographic composition principles such as the

rule of thirds, diagonal dominance and visual balance [Liu et al.,] which, for the moment only considers *straight* lines.

Sculpting motion

The wave curves in Chapter 5 is one example of an intuitive "motion stroke"—a static drawing that encodes a dynamic motion. We could very well imagine different shapes and gestures translating into different motions that could be used to *sculpt* or *paint* the motion. One of the limitations of the geometric methods used in Chapter 5 (i.e. warping and blending) is that it is difficult to combine and weigh different constraints. For example, a stylistic dance may require a wave to pass through the character's arms, but his hand may also require remaining fixed at some point in time. Combining these constraints—matching the wave, while holding fixed positions—could be realized in a constrained optimization framework. We could envision a system where the DLOA seeks to match the shape of the dynamic wave curve as much as possible, while satisfying hard position constraints in time. Additional motion strokes could be painted on top and mixed with the underlying wave motion where the weight of each constraint would be given by the amount of re-drawing and stylus pressure exerted by the artist.

Storyboard to animation

The premise of this thesis is that high-level artistic goals are indirectly controlled with current animation tools. Telling stories with the body language of a character requires tools that allow indicating motions and viewpoints at a much larger time scale than when designing specific actions. It would be interesting to investigate how storyboard annotations can be used to establish character motions.

This may also require inventing new sketch-based annotations. For instance, the interaction between characters is an important aspect of narratives that could be specified in a storyboard-like interface. How to naturally specify interactions is an very interesting area of research where sketching could be prove beneficial. For example, two characters holding hands can be sketched with a single stroke, as in show in Fig. 3-12 from chapter 3. Another idea is how lines of action often "move out" of collision points. Hence specifying a collision point could automatically generate a visible line of action for posing a character.

Aesthetically pleasing physics-based controllers

Generative motion is a very important area of research for interactive games. The ability to automatically and realistically react to the environment enriches the immersion in games which currently use pre-defined (and repetitive) animations for interactions. As physics-based controllers become more robust and versatile, they will start appearing in games and virtual worlds. Eventually, we will want the character to not only realistically perform actions, but have its own style and exhibit aesthetically pleasing poses whenever possible. The line of action constraint presented in Chapter 3 could be used as objective with online physics-based controllers to steer the movement towards aesthetically pleasing shapes for the viewer by deriving virtual forces from the image space constraint.

Bibliography

[Car, 1994] (1994). Cartoon physics. *Cartoon Physics*.

[Pri, 2015] (2015). Principles of art. *wikipedia: principles of art*.

[Abe et al., 2004] Abe, Y., Liu, C. K., Hertzmann, A., and Popović, Z. (2004). Momentum-based parameterization of dynamic character motion. In *ACM SIGGRAPH/Ehrographics Symposium on Computer Animation*, pages 173–182.

[Abling, 2012] Abling, B. (2012). *Fashion Sketchbook*. Fairchild Publications.

[Alexa et al., 2000] Alexa, M., Cohen-Or, D., and Levin, D. (2000). As-rigid-as-possible shape interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 157–164, New York, NY, USA.

[Arikan and Forsyth, 2002] Arikan, O. and Forsyth, D. A. (2002). Interactive motion generation from examples. *ACM Trans. Graph.*, 21(3):483–490.

[Autodesk, 2015] Autodesk (2015). Maya. `http://www.autodesk.com/maya`.

[Blair, 1994] Blair, P. (1994). *Cartoon Animation*. Walter Foster.

[Borno et al., 2013] Borno, M. A., de Lasa, M., and Hertzmann, A. (2013). Trajectory optimization for full-body movements with complex contacts. *IEEE Trans. Vis. Comput. Graph.*, 19(8):1405–1414.

[Brand and Hertzmann, 2000] Brand, M. and Hertzmann, A. (2000). Style machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 183–192, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Bregler et al., 2002] Bregler, C., Loeb, L., Chuang, E., and Deshpande, H. (2002). Turning to the masters: motion capturing cartoons. *ACM Transactions on Graphics*, 21(3):399–407.

[Brooks and Pilcher, 2001] Brooks, B. and Pilcher, T. (2001). *The Complete Cartooning Course*. David & Charles Publishers.

[Burtnyk and Wein, 1976] Burtnyk, N. and Wein, M. (1976). Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *ACM Transactions on Graphics (TOG)*, 19(10):564–569.

[Chai and Hodgins, 2005] Chai, J. and Hodgins, J. K. (2005). Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696.

[Chai and Hodgins, 2007] Chai, J. and Hodgins, J. K. (2007). Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics (TOG)*, 26.

[Chen et al., 2012] Chen, J., Izadi, S., and Fitzgibbon, A. (2012). Kinetre: Animating the world with the human body. In *In Proc. UIST*, UIST '12, New York, NY, USA. ACM.

[Choi et al., 2012] Choi, M. G., Yang, K., Igarashi, T., Mitani, J., and Lee, J. (2012). Retrieval and visualization of human motion data via stick figures. *Computer Graphics Forum*, 31(7):2057–2065.

[Cohen, 1992] Cohen, M. F. (1992). Interactive spacetime control for animation. pages 293 – 302.

[Coleman et al., 2008] Coleman, P., Bibliowicz, J., Singh, K., and Gleicher, M. (2008). Staggered poses: a character motion representation for detail-preserving editing of pose and coordinated timing. pages 137–146.

[Coros et al., 2010] Coros, S., Beaudoin, P., and van de Panne, M. (2010). Generalized biped walking control. *ACM Trans. Graph.*, 29(4):130:1–130:9.

[Davis et al., 2003] Davis, J., Igarashi, M., Chuang, E., Popovic', Z., and Salesin, D. (2003). A sketching interface for articulated figure animation. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 320–328.

[Davis et al., 2008] Davis, R. C., Colwell, B., and Landay, J. A. (2008). K-sketch: A 'kinetic' sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 413–422, New York, NY, USA. ACM.

[de Lasa et al., 2010] de Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. *ACM Trans. Graph.*, 29(4):131:1–131:10.

[DLIB,] DLIB. Optimization library.

[Dontcheva et al., 2003] Dontcheva, M., Yngve, G., and Popovic', Z. (2003). Layered acting for character animation. In *SIGGRAPH*, pages 409–416.

[Doucet, 2011] Doucet, R. (2011). Line of action-mickey's christmas carol. *http://floobynooby.blogspot.fr/2011/01/lines-of-action-mickeys-christmas-carol.html*.

[Esposito et al., 1995] Esposito, C., Paley, W. B., and Ong, J. (1995). Of mice and monkeys: A specialized input device for virtual body animation. In Zyda, M., editor, *SI3D*, pages 109–114, 213. ACM.

[Felzenszwalb and Huttenlocher, 2005] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1):55–79.

[Foundation, 2015] Foundation, B. (2015). Blender. *https://www.blender.org/*.

[Gleicher, 1997] Gleicher, M. (1997). Motion editing with spacetime constraints. pages 139–148.

[Gleicher, 2001] Gleicher, M. (2001). Motion path editing. In *Proceedings 2001 ACM Symposium on Interactive 3D Graphics*. ACM.

[Grochow et al., 2004] Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531.

[Hahn et al., 2012] Hahn, F., Martin, S., Thomaszewski, B., Sumner, R., Coros, S., and Gross, M. (2012). Rig-space physics. *ACM Trans. Graph.*, 31(4):72:1–72:8.

[Hämäläinen et al., 2014] Hämäläinen, P., Eriksson, S., Tanskanen, E., Kyrki, V., and Lehtinen, J. (2014). Online motion synthesis using sequential monte carlo. *ACM Trans. Graph.*, 33(4):51:1–51:12.

[Hart, 1997] Hart, C. (1997). *How to draw animation*. Watson Guptill.

[Hodgins et al., 1995] Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 71–78, New York, NY, USA. ACM.

[Howe et al., 2000] Howe, N. R., Leventon, M. E., and Freeman, W. T. (2000). Bayesian reconstruction of 3d human motion from single-camera video. *Advances in Neural Information Processing Systems*, 10.

[Hsu et al., 2005] Hsu, E., Pulli, K., and Popovic', J. (2005). Style translation for human motion. *ACM Transactions on Graphics*, 24:1082–1089.

[Huang et al., 2013] Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., and Chen, B. (2013). L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4):65:1–65:8.

[Igarashi et al., 1998] Igarashi, T., Kadobayashi, R., Mase, K., and Tanaka, H. (1998). Path drawing for 3d walkthrough. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, pages 173–174, New York, NY, USA. ACM.

[Igarashi et al., 1999] Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: a sketching interface for 3d freeform design. *ACM Transactions on Graphics (TOG)*, pages 409–416.

[Igarashi et al., 2005] Igarashi, T., Moscovich, T., and Hughes, J. F. (2005). As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141.

[Jacobson et al., 2012] Jacobson, A., Baran, I., Kavan, L., Popović, J., and Sorkine, O. (2012). Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4).

[Jacobson et al., 2014] Jacobson, A., Panozzo, D., Glauser, O., Pradalier, C., Hilliges, O., and Sorkine-Hornung, O. (2014). Tangible and modular input device for character articulation. *ACM Trans. Graph.*, 33(4):82:1–82:12.

[Karpenko and Hughes, 2006] Karpenko, O. A. and Hughes, J. F. (2006). Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (TOG)*, 25:589–598.

[Kho and Garland, 2005] Kho, Y. and Garland, M. (2005). Sketching mesh deformations. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pages 147–154, New York, NY, USA. ACM.

[Kirk et al., 2005] Kirk, A. G., O'Brien, J. F., and Forsyth, D. A. (2005). Skeletal parameter estimation from optical motion capture data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '05*, pages 782–788.

[Knep et al., 1995] Knep, B., Hayes, C., Sayre, R., and Williams, T. (1995). Dinosaur input device. In *CHI*, pages 304–309.

[Kovar et al., 2002] Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Trans. Graph.*, 21(3):473–482.

[Krishna M., 2012] Krishna M., S. (2012). Learn to draw dynamic comic characters using the "two can" technique! *Blog post*.

[Lasseter, 1987] Lasseter, J. (1987). Principals of traditional animation applied to 3d computer animation. *ACM SIGGRAPH Computer Graphics*, 21(4):35–44.

[Laszlo et al., 2000] Laszlo, J., van de Panne, M., and Fiume, E. (2000). Interactive control for physically-based animation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 201–208, New York, NY, USA.

[Lau et al., 2009] Lau, M., Bar-Joseph, Z., and Kuffner, J. (2009). Modeling spatial and temporal variation in motion data. *ACM Trans. Graph.*, 28(5):171:1–171:10.

[Laybourne, 1998] Laybourne, K. (1998). *The Animation Book: A Complete Guide to Animated Filmmaking, from Flip-Books to Sound Cartoons*. Three Rivers Press.

[Lee and Chen, 1985] Lee, H.-J. and Chen, Z. (1985). Determination of 3d human body postures from a single view. *Computer Vision, Graphics, and Image Processing*, 29(3):396.

[Lee and Shin, 1999] Lee, J. and Shin, S. Y. (1999). A hierarchical approach to interactive motion editing for human-like figures. pages 39–48.

[Lee and Buscema, 1978] Lee, S. and Buscema, J. (1978). *How To Draw Comics The Marvel Way*. Simon Schuster Inc.

[Lin et al., 2010] Lin, J., Igarashi, T., Mitani, J., and Saul, G. (2010). A sketching interface for sitting pose design. pages 111–118.

[Lipman et al., 2008] Lipman, Y., Levin, D., and Cohen-Or, D. (2008). Green coordinates. *ACM Trans. Graph.*, 27(3).

[Liu et al., 2006] Liu, C. K., Hertzmann, A., and Popović, Z. (2006). Composition of complex optimal multi-character motions. In *ACM SIGGRAPH/Ehrographics Symposium on Computer Animation*.

[Liu and Popović, 2002] Liu, C. K. and Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animation. *ACM Trans. on Graphics (SIGGRAPH 2002)*.

[Liu et al.,] Liu, L., Chen, R., Wolf, L., and Cohen-Or, D. Optimizing photo composition.

[Liu et al., 2013] Liu, L., Yin, K., Wang, B., and Guo, B. (2013). Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph.*, 32(6):215:1–215:8.

[Ma et al., 2010] Ma, W., Xia, S., Hodgins, J. K., Yang, X., Li, C., and Wang, Z. (2010). Modeling style and variation in human motion. In *Symposium on Computer Animation (SCA)*, pages 21–30.

[Mao et al., 2005] Mao, C., Qin, S. F., and Wright, D. (2005). A sketch-based gesture interface for rough 3d stick figure animation. *Proceedings of the 2005 Eurographics workshop on Sketch-Based Interfaces and Modeling*.

[Min et al., 2009] Min, J., Chen, Y.-L., and Chai, J. (2009). Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics (TOG)*, 29.

[Mordatch et al., 2012] Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4).

[Muico et al., 2009] Muico, U., Lee, Y., Popović, J., and Popović, Z. (2009). Contact-aware nonlinear control of dynamic characters. In *ACM SIGGRAPH 2009*, SIGGRAPH '09, pages 81:1–81:9, New York, NY, USA. ACM.

[Neff et al., 2007] Neff, M., Albrecht, I., and Seidel, H.-P. (2007). Layered performance animation with correlation maps. *Comput. Graph. Forum*, 26(3):675–684.

[Neff and Fiume, 2003] Neff, M. and Fiume, E. (2003). Aesthetic edits for character animation. pages 239–244.

[Ngo and Marks, 1993] Ngo, J. T. and Marks, J. (1993). Spacetime constraints revisited. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 343–350, New York, NY, USA. ACM.

[Noble and Tang, 2006] Noble, P. and Tang, W. (2006). Automatic expressive deformations for stylizing motion. *GRAPHITE '06 Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 57–63.

[O'Brien et al.,] O'Brien, J. F., Bodenheimer, R. E., Brostow, G. J., and Hodgins, J. K. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface*, GI '00.

[Oore et al., 2002] Oore, S., Terzopoulos, D., and Hinton, G. (2002). A desktop input device and interface for interactive 3d character animation. In *In Proc. Graphics Interface*, pages 133–140.

[Öztireli et al., 2013] Öztireli, A. C., Baran, I., Popa, T., Dalstein, B., Sumner, R. W., and Gross, M. (2013). Differential blending for expressive sketch-based posing. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM.

[Pullen and Bregler, 2002] Pullen, K. and Bregler, C. (2002). Motion capture assisted animation: Texturing and synthesis. *ACM Trans. Graph.*, 21(3):501–508.

[Rhodin et al., 2014] Rhodin, H., Tompkin, J., In Kim, K., Varanasi, K., Seidel, H. P., and Theobalt, C. (2014). Interactive motion mapping for real-time character control. *Computer Graphics Forum*, 33(2):273–282.

[Riki, 2013] Riki, J. (2013). How fast you animate. *URL Link*.

[Rohmer et al., 2009] Rohmer, D., Hahmann, S., and Cani, M.-P. (2009). Exact volume preserving skinning with shape control. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, SCA '09, pages 83–92. ACM.

[Rose et al., 1998] Rose, C., Bodenheimer, B., and Cohen, M. F. (1998). Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications*, 18:32–40.

[Safonova et al., 2004] Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 514–521, New York, NY, USA. ACM.

[Sederberg and Greenwood, 1992] Sederberg, T. W. and Greenwood, E. (1992). A physically based approach to 2-d shape blending. In *Computer Graphics*, volume 26, pages 25–34.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. In *In Proc. SIGGRAPH 86*, pages 151–160.

[Seol et al., 2013] Seol, Y., O'Sullivan, C., and Lee, J. (2013). Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 213–221, New York, NY, USA. ACM.

[Shapiro et al., 2006] Shapiro, A., Cao, Y., and Faloutsos, P. (2006). Style components. In *Proceedings of Graphics Interface 2006*, GI '06, pages 33–39.

[Shapiro and Lee, 2009] Shapiro, A. and Lee, S.-H. (2009). Practical character physics for animators. In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pages 55:1–55:1, New York, NY, USA. ACM.

[Shiratori et al., 2013] Shiratori, T., Mahler, M., Trezevant, W., and Hodgins, J. K. (2013). Expressing animated performances through puppeteering. In *3DUI*, pages 59–66. IEEE.

[Shotton et al., 2011] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in

parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA. IEEE Computer Society.

[Sorkine et al., 2004] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA. ACM.

[Sumner and Popović, 2004] Sumner, R. W. and Popović, J. (2004). Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405.

[Taylor, 2000] Taylor, C. J. (2000). Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Underst.*, 80(3):349–363.

[Thorne et al., 2004] Thorne, M., Burke, D., and van de Panne, M. (2004). Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics (TOG)*, 23:424–431.

[Toribash, 2015] Toribash (2015). *http://www.toribash.com/*.

[Torresani et al., 2006] Torresani, L., Hackney, P., and Bregler, C. (2006). Learning motion style synthesis from perceptual observations. In *NIPS*, pages 1393–1400. MIT Press.

[Unuma et al., 1995] Unuma, M., Anjyo, K., and Takeuchi, R. (1995). Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95.

[Wampler et al., 2014] Wampler, K., Popović, Z., and Popović, J. (2014). Generalizing locomotion style to new animals with inverse optimal regression. *ACM Trans. Graph.*, 33(4):49:1–49:11.

[Wang et al., 2006] Wang, J., Drucker, S. M., Agrawala, M., and Cohen, M. F. (2006). The cartoon animation filter. *ACM Trans. Graph.*, 25(3):1169–1173.

[Wei and Chai, 2011] Wei, X. and Chai, J. (2011). Intuitive interactive human character posing with millions of example poses. In *IEEE Comput. Graph. Appl.*, volume 31, pages 78–88.

[Williams, 2002] Williams, R. (2002). *The Animator's Survival Kit*.

[Witkin and Kass, 1988] Witkin, A. and Kass, M. (1988). Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 159–168, New York, NY, USA. ACM.

[Wu and Popović, 2010] Wu, J.-c. and Popović, Z. (2010). Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph.*, 29(4):72:1–72:10.

[Yamane et al., 2010] Yamane, K., Ariki, Y., and Hodgins, J. (2010). Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 169–178.

[Ye and Liu, 2010] Ye, Y. and Liu, C. K. (2010). Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010*, SIGGRAPH '10.

[Yin et al., 2007] Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3).

[Yoo et al., 2014] Yoo, I., Vanek, J., Nizovtseva, M., Adamo-Villani, N., and Benes, B. (2014). Sketching human character animations by composing sequences from large motion database. *The Visual Computer*, 30(2):213–227.

[Yoshizaki et al., 2011] Yoshizaki, W., Sugiura, Y., Chiou, A. C., Hashimoto, S., Inami, M., Igarashi, T., Akazawa, Y., Kawachi, K., Kagami, S., and Mochimaru, M. (2011). An actuated physical puppet as an input device for controlling a digital manikin. In *CHI*, pages 637–646.

[Zhao and Badler, 1994] Zhao, J. and Badler, N. (1994). Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13:313–336.

[Zheng et al., 2015] Zheng, Q., Hao, Z., Huang, H., Xu, K., Zhang, H., Cohen-Or, D., and Chen, B. (2015). Skeleton-intrinsic symmetrization of shapes. *Computer Graphics Forum (Proc. of Eurographics)*.