OptiMo: Optimization-Guided Motion Editing for Keyframe Character Animation

Yuki Koyama **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{ koyama.y, m.goto }@aist.go.jp

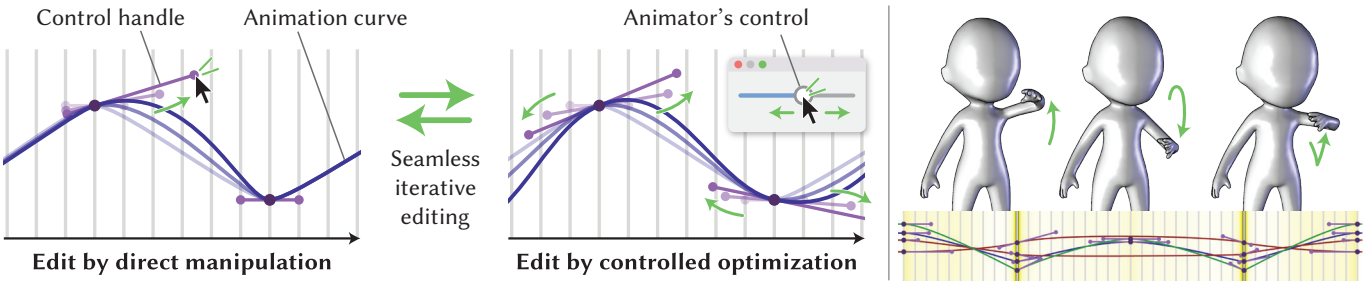

Figure 1. We present a new framework for animators to edit character motions by effectively using the power of numerical optimization. (Left) Concept of the framework. As well as direct manipulation, animators can use controlled optimization to efficiently edit animation curves in the iterative editing process. (Right) Example of edited motions using our proof-of-concept system, named *OptiMo*.

ABSTRACT

The mission of animators is to create nuanced, high-quality character motions. To achieve this, the careful editing of *animation curves*—curves that determine how a series of keyframed poses are interpolated over time—is an important task. Manual editing affords full and precise control, but requires tedious and nonintuitive trials and errors. Numerical optimization can automate such exploration; however, automatic solutions cannot always be perfect, and it is difficult for animators to control optimization owing to its black-box behavior. In this paper, we present a new framework called *optimization-guided motion editing*, which is aimed at maintaining a sense of full control while utilizing the power of optimization. We have designed interactions and developed a set of mathematical formulations to enable them. We discuss the framework's potential by demonstrating several usage scenarios with our proof-of-concept system, named *OptiMo*.

ACM Classification Keywords

H.5.m Information Interfaces and Presentation (e.g. HCI): Miscellaneous; I.3.8 Computer Graphics: Applications

Author Keywords

Motion editing; computer animation; optimization; computational design; physics.

INTRODUCTION

In the digital production of animated films, computer games, *etc.*, the mission of animators is to animate computer graphics characters plausibly and appealingly. Their task could be decomposed into two stages: *keyframing* and *curve editing*. Keyframing is the stage in which animators define *key poses* at several time points (*i.e.*, *keyframes*). Curve editing is the stage in which animators manipulate *animation curves* to define how key poses are interpolated between keyframes. Whereas keyframing is important as it defines the basic motion scenario, curve editing is also important for achieving well nuanced animations; animations using the same keyframing can produce very different impressions, depending on how the animation curves are edited.

Manually creating plausible animations requires tedious and unintuitive trials and errors of animation curve editing. It requires to explore a high-dimensional search space by manipulating many control handles (note that a human pose is already high-dimensional and a human motion is even higher-dimensional). Additionally, it is difficult to predict the quality before actually modifying the animation curves and seeing the new motion, because the effect of manipulating a parameter is often indirect. Furthermore, plausible animation requires adherence to certain high-level rules such as physics, which makes the task even more complex.

Numerical optimization is a powerful tool and has the potential to fully automate the process of animation curve editing. By mathematically formulating the search space (*i.e.*, which parameters should be adjusted) and the objective (*i.e.*, how the motion should be) of the curve editing, an optimizer can automatically and simultaneously optimize the target parameters, which are otherwise manipulated one-by-one manually.

This approach could release animators from the difficulty of handling the high-dimensional problem.

However, automatic solutions cannot always be perfect because any objective function cannot perfectly represent animators' aesthetic intention. Thus, it is inevitable that the resulting animation needs to be manually edited by animators. However, owing to the black-box nature of optimization, editing the optimized animation is often much more difficult than editing the original one; animators have to begin by understanding what happened with the optimization. Furthermore, some optimization may change the representation of animation curves (*e.g.*, the manner of placing control handles for editing curves), which makes it more painful to understand and edit them. Thus, animators are rarely comfortable relying on such a black box, and so the manual approach is mainly adopted in practice.

Contributions

In this paper, we present a new framework for editing articulated character motions, which we call *optimization-guided motion editing*, where optimization is used as a tool for effectively guiding animators' manual motion editing, rather than for fully automating the process (Figure 1). This framework has the advantages of both automatic optimization and manual editing; it borrows the power of numerical optimization to manipulate multiple parameters simultaneously, yet tries to give animators the sense of full control. The objective used for optimization cannot be perfect but can be an "approximation" of what animators want to create, so it can guide motion editing toward generally better directions, especially at the early stage of exploration. This would help expert animators save time and allow novices to more easily achieve high-quality animations.

Our primary contributions are the design and mathematical formulations for a proof-of-concept system, named *OptiMo*, which enables optimization-guided motion editing. Specifically, to keep the sense of full control as much as possible while interacting with optimization, we considered the following three design goals for the system (Figure 2):

Editability. The optimized motion should be able to be easily edited by animators for further manual refinement.

Controllability. The optimization behavior should be able to be controlled by animators as flexibly as possible.

Transparency. The optimization process should be understandable for animators as intuitively as possible.

To achieve these goals, we developed the following features for our system:

- It solves optimization problems without changing the placement of the control handles that the animator defined in the keyframing stage. This is useful for preserving editability and for enhancing transparency.
- It allows animators to control the optimization behavior via simple interface (*e.g.*, sliders). To further increase controllability, some optimization controls are designed to be interactive; animators can intervene in the optimization process during running it.
- It visualizes certain aspects of the optimization for enhancing transparency. This includes the visualization of the optimization process (not just the result) and the visualization of how the current motion is evaluated by the optimizer.

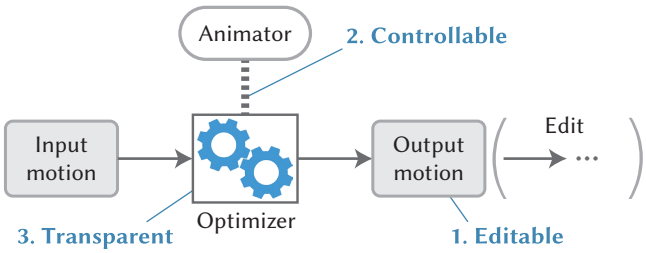

Figure 2. Illustration of our design goals: (1) the output motion should remain editable for further refinement, (2) the optimization should be controllable by the animator, and (3) the process should be transparent.

We discuss the potential of optimization-guided motion editing through demonstrating several usage scenarios using our proof-of-concept system. We also report the feedback comments from expert animators obtained through an informal interview.

RELATED WORK

Character Motion Editing

Character animations are usually created by either captured or keyframed motions. For realistic humanoid characters, motion capture is mainly used because it is effective for creating realistic motions. One common problem with this approach is the difficulty of editing motions once they are captured, although researchers have tackled this problem (*e.g.*, [33, 27]). Apart from the editability issue, there are many cases in which motion capture is not effective. First, captured motions are often unsuitable for stylized characters (*e.g.*, those in Pixar's feature films) because the mismatch of body balance may cause perceptual incongruity and captured motions may be too realistic for stylized characters. For non-humanoid characters (*e.g.*, creatures), it is not easy to capture their motions or apply captured motions performed by human actors to them. Additionally, it is difficult to capture highly acrobatic motions.

Thus, for such cases in which motion capture is not effective, animators create motions from scratch, which is our target scenario. Using dedicated systems such as Maya [2], animators create key poses at several key frames, and then adjust how they are interpolated by editing the animation curves. Both stages involve adjustment of many parameters, and so require expertise for exploring such a high-dimensional search space. Researchers have developed systems for supporting this task; for example, sketch-based interaction can help animators intuitively explore this search space [16, 8]. In this work, we investigate the use of optimization to support this exploration.

Several works focused on the creation of key poses. Tangible devices have been developed for setting key poses of various characters [13, 19]. Some methods enable posing by sketching *line-of-actions* [29] or *gesture drawings* [6]. Our framework focuses on the stage of curve editing, which comes after the

posing stage, and so setting key poses is out of our scope. Note that our framework is complementary to these techniques.

Editing animation curves plays an important role in determining the quality of motions, but few studies have tackled this aspect. Rather than supporting interactive curve editing, automatic synthesis of interpolation curves between keyframes has been investigated. This approach is called *space-time constraints* [37, 14]. However, this approach is not preferred in practical workflow because generated motions are difficult to edit. In contrast to this, we aim at achieving interactive exploration by animators with the support of iterative editing.

A method, called *rig-space physics* [17], is similar to ours in terms of maintaining the editability of resulting motions. It generates secondary motions for passive parts (*e.g.*, a fat character's belly shaking because of a primary motion, such as walking) by using elastic simulation. This simulation is performed using artist-defined *rig* parameters; any deformation is represented as a combination of rig parameters, enabling intuitive editing by animators after generation. In the same spirit, but taking a different approach, our framework not only preserves the original rigs (*i.e.*, we do not change skeletal structures) but also preserves the control handles of the animation curves defined in the keyframing stage. Additionally, we target characters' active motions rather than their passive ones, and we use optimization to support edits rather than physical simulation to generate new motions.

Shapiro and Lee [32] presented an animation authoring system that guides animators based on visualization of characters' physical properties (*e.g.*, the center of mass and angular momentum). The motivation and the system design are overlapped to ours; whereas using a high-level concept (physics in their case and optimization in our case), it allows animators to maintain "*a fine-grained level of control*" unlike other automatic methods to achieve practicality in actual production. Whereas their system only provides visualization and does not manipulate any parameters automatically, our framework semi-automatically manipulates many parameters with the control of animators, which is more challenging for keeping the control, and thus requires non-trivial interaction design. Note that their focus is the support of editing whole-body movement paths (*e.g.*, matching the path of the center of mass with a ballistic trajectory) and it does not explicitly guide how the character's poses and timings should be nuanced. In contrast, our framework supports it via optimizing animation curves.

Some methods support editing and authoring of two-dimensional (2D) animations [22, 9]. Although our work is primarily targeted at three-dimensional (3D) animations, where the degrees of freedom are often much larger than 2D, we believe that the concept and interactions of optimization-guided motion editing could be transferred to 2D cases, which we leave as a future work.

Optimization-Based Design

Researchers have shown many design applications in which numerical optimization enables new designs or new design workflows that are otherwise difficult. This paradigm is often referred to as *computational design*. Some methods are fully automatic; for example, spinning top design [4] and keyboard layout design [21] can be automatically generated once the user specifies necessary inputs.

Other methods involve interaction with optimizers during design sessions, and ours fits into this category. Local optimization (*e.g.*, the steepest descent method) can be used for gradually guiding a design toward better directions even during user manipulation [23, 35]. Optimization can also be used for generating alternative designs as suggestions; DesignScape [28] and Sketchplore [34] are systems for authoring graphic designs with optimized layout suggestions. From the viewpoint of how users interact with optimization, MenuOptimizer [5] has several similarities to our approach, although the target domain is quite different; the system offers visualization of the metrics and control of the objective function. Optimization is also useful for converting physically infeasible inputs into feasible ones; Airways [12] uses optimization for designing physically feasible quadrotor trajectories from casual sketches. Our system adopts a physics-based cost function that is minimized in the optimization and thus can guide target motions in physically plausible directions, as explained later.

Our work contributes to this research area by presenting the first system for optimization-guided design in the motion editing scenario. We investigate the necessary interaction design and mathematical formulations for this specific scenario.

TARGET PROBLEM

Character Representation

Characters in computer animation are articulated by controlling the *animation rig*, which is a generic term for indicating a set of artist-defined variables that parametrizes the character's pose [17, 18]. Several types of rig parameters are used together in the production process; for example, *blendshapes* is often used for manipulating relatively detailed poses such as facial expressions [25]. Among them, *skeletal rig* is probably the most basic rig system, in which the pose of a character is controlled by a tree structure called a *skeleton*: a set of *bones* connected by *joints*. In this study, we test the concept of optimization-guided motion editing using skeletal animation, because it is simple and widely used.

By assuming that bones are rigid (*i.e.*, do not change their lengths), a character's pose is fully specified by the orientations of the joints. Each joint orientation is controlled by either *Forward Kinematics* (FK) or *Inverse Kinematics* (IK). In FK, the orientation of a joint (specified by Euler angles in our implementation) is considered as the rig parameter that should be tweaked by animators. In IK, the target position of a joint, called the *end effector*, is considered as the rig parameter; in this case, the orientations of intermediate joints are derived by solving IK [7]. See Figure 3 (Left) for an example. Usually, a character is controlled by using both FK and IK; for example, spine and neck are often controlled by FK, and arms and legs are often controlled by IK with hand or foot target positions.

Animation Curve Representation

An animation curve interpolates keyframed rig values (*i.e.*, pairs of a rig value and a time point) over time. For this,

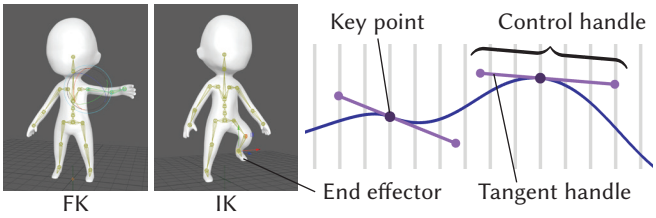

Figure 3. (Left) Example of a skeletal rig. The skin mesh is bound to the internal skeleton. Animators manipulate either the joint angles in forward kinematics (FK) or the end effector positions in inverse kinematics (IK). (Right) An animation curve and the terms used in this paper.

we chose the representation based on piecewise cubic Bézier curves because it is widely used in animation software [2, 1]. The entire curve passes through several key points. A partial curve between two key points is called *curve segment*, which is a single cubic Bézier curve. We define a *control handle* to be a manipulatable handle consisting of a *key point* (a pair of a rig value and a time point) and a *tangent handle* (Figure 3; Right). Key points determine the end points of each curve segment, and tangent handles control the curves shapes. Defining key points corresponds to the keyframing stage, and manipulating tangent handles corresponds to the curve editing stage.

Task Goal

The goal of curve editing is to make the motion plausible and appealing by adjusting the tangent handles. This is usually done by manual one-by-one iterations of the following process: manipulate a tangent handle, watch the motion, and go to the next tangent handle until the motion becomes satisfactory. This task is not only tedious but also complicated owing to the high-dimensionality and the indirect effects among rig parameters. For example, suppose that a humanoid character is waving his arm. Even if the shoulder joint is the primary moving part, it affects other body parts to keep the balance from the viewpoint of physics, suggesting that even a simple motion requires to consider the indirect effects.

OPTIMIZATION-GUIDED MOTION EDITING

Design Goals

We propose to utilize numerical optimization as a tool for guiding manual motion editing. However, naïvely applying optimization to a target motion is not desirable because it deprives animators of full and precise control. To maintain a sense of full control as much as possible while interacting with optimization, we set the following design goals (Figure 2):

The optimization output should remain editable. The resulting motion should be easily editable by animators. This enables further refinement, which is crucial in practical workflow. In particular, it is extremely important not to change the control handle definitions before and after the optimization. This is because every control handle is set (more or less) "intentionally" by the animator in the keyframing stage, and thus the placement of the control handles before optimization could be the most convenient representation for the animator who performed the keyframing.

The optimization behavior should be controllable. Animators should be able to control the behavior of optimization as freely and precisely as possible. To this end, the system must provide ways for animators to intervene in the optimization process, possibly by tweaking several control parameters. Additionally, it is desirable for the intervention to be interactive during running the optimization, thereby allowing fine-grained adjustment.

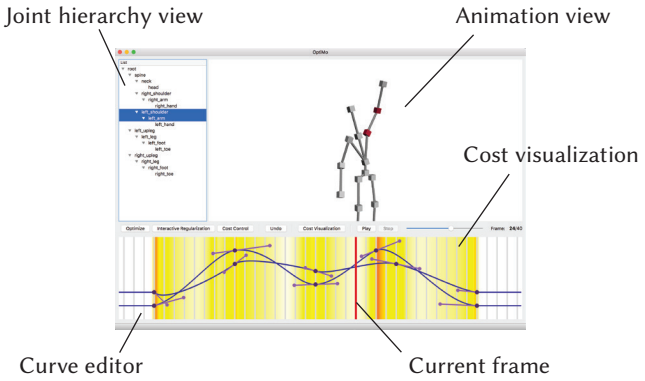

Figure 4. Appearance of OptiMo, a proof-of-concept system for optimization-guided motion editing.

The optimization process should be transparent. Optimization is an abstract concept and it may be difficult for those who are not familiar with mathematics to understand *why* it works. However, it is necessary for animators to understand *how* it works; otherwise, animators cannot effectively decide when and how to utilize the optimization during editing. To this end, the system must provide information about what the optimization will do or is doing in an easily understandable way.

OptiMo: A Proof-of-Concept System

Based on the above considerations, we designed and developed a proof-of-concept system of optimization-guided motion editing, named OptiMo (Figure 4). Similar to most motion authoring software, our system has a *curve editor*[1] in its interface, in which the rig parameters of the selected joints or IK handles are shown as animation curves, and animators can edit the curves by manipulating the control handles. The selection of joints and IK handles is done via the *joint hierarchy view*. At any time, animators can trigger optimization for improving the animation curves. Each optimization finishes typically in a few seconds or up to ten seconds, depending on motions. Once the optimization is done, animators can immediately begin manual refinement or run the optimization again with a different control setting. For better interaction with optimization, the system was designed to have the following features.

Preservation of Control Handles (Editability + Transparency)

The system preserves the placement of the animator-defined control handles during and after the optimization; the optimization adjusts the tangent handles and does not touch the key points defined in the keyframing stage. It does not change the curve representation, nor re-define control handles automatically. This allows animators to easily understand what the

[1]This widget is sometime called by different names, such as *Graph Editor* in Maya [2] and *FCurves Window* in MotionBuilder [3].

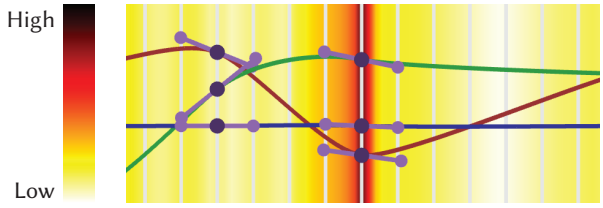

Figure 5. Cost visualization on the curve editor using a color map. High-cost regions (around the center in this figure) indicate that the system considers the current motion to be undesirable around there, and the optimization will try to resolve these undesirable moments.

optimization did, and thus allows them to immediately begin to edit the resulting curves for fine-grained refinement or undo the optimization if necessary.

Optimization Process Visualization (Transparency)
Not only does it provide the output of the optimization, the system also visualizes the intermediate states of the optimization as real-time animation in the curve editor. That is, animators can see how the optimization explores possible motions and searches for a solution. Furthermore, as our system preserves the control handles during the optimization and manipulates the handles that animators otherwise manipulate, animators can understand the visualization as a "fast forward" of a manual search performed by someone else. This allows animators to understand the concept of optimization, which increases transparency. This animated visualization also helps animators easily notice unintentional modifications (if any).

Cost Visualization (Transparency)
The optimization process visualization facilitates understanding of how the optimization finds a solution; however, it does not tell why the solution is chosen. To make it transparent, the system shows the distribution of *cost* values over time in the curve editor using a color map (Figure 5). This distribution indicates what the system considers "good" or "bad;" by optimization, the system tries to minimize the total "badness" in the motion. As explained later, we adopted a physics-based cost function [31] to quantify the goodness of motions.

Interactive Regularization (Controllability + Transparency)
Naïve optimization sometimes modifies the motion so much that it might break the original intention. To prevent this, the system allows animators to "regularize" the optimization behavior, or to control how much the optimization preserves the original motion. This can be done by adjusting a single *regularization weight* parameter via a slider interface; a larger weight makes the optimization more conservative, and *vice versa*. Furthermore, the system allows interactive specification of the weight parameter *during* optimization; according to the slider manipulation, the curves and the preview motion are updated in real time using the regularized optimization. This interactivity supports animators not only to control the optimization easily and precisely, but also to understand how the regularization works. Figure 6 illustrates this feature.

Per-Joint Regularization (Controllability)
The system allows animators to control the strength of regularization for each joint separately. For example, if the animator is not satisfied with the result of an optimization because some specific joints are excessively modified from the original motion, he or she can undo the optimization, set a higher regularization weight for each of these specific joints, and then re-run optimization to obtain a more desirable result. Figure 7 illustrates an example of the usage of this feature.

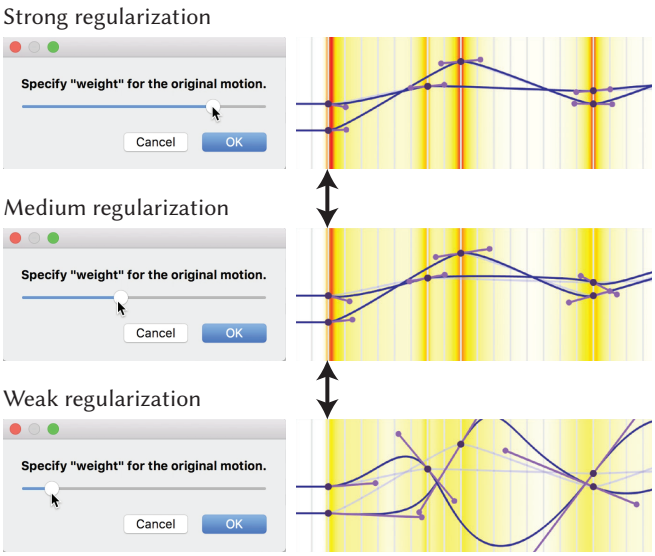

Figure 6. Interactive specification of regularization weight. The original motion is more preserved when a larger weight is specified, and *vice versa*. By the real-time preview of the optimized curves and motions, the user can intuitively seek a preferable weight value for controlling the optimization behavior.

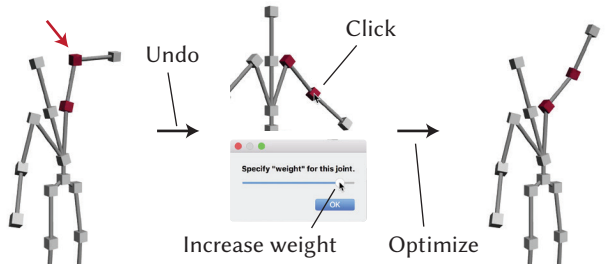

Figure 7. Per-joint regularization. In this example, the animator is unsatisfactory with the result of optimization because the elbow is modified in an undesirable manner (indicated by a red arrow). This can be fixed by undoing the failed optimization, setting a large regularization weight for the elbow joint, and running the optimization again.

Time-Varying Cost Control (Controllability)
Because no cost function is perfect, the visualized cost distribution may be unsatisfactory for the animator; it can provide unintentionally high cost values for certain moments. Applying optimization with this unsatisfactory cost distribution results in a motion that does not correctly reflect the animator's intention. To prevent this, the system has a feature called *time-varying cost control*; it allows animators to specify a *time-varying weight distribution*, which is a smooth distribution of weight values over time, to tell the system to ignore costs in low-weight periods. Figure 8 illustrates this feature. Furthermore, to make adjustments to the weight distribution easier, there is an interactive mode; it can run optimization in real time during adjustment.

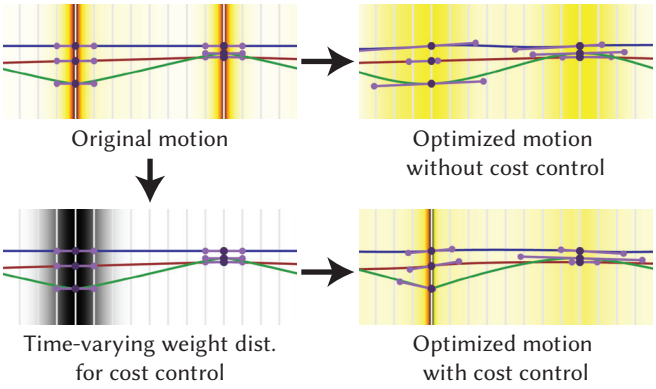

Figure 8. Time-varying cost control. If the visualized cost distribution does not correctly reflect the intention at some periods, the user can control where to ignore the cost function by specifying a time-varying weight distribution. Here, the darker regions in the curve editor mean lower weights, and thus the costs of these regions will be ignored in the optimization.

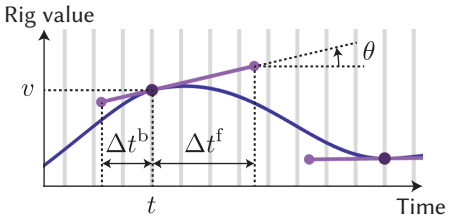

Figure 9. Illustration of the parameters of a control handle. The time t and the value v are provided at the keyframing stage. The task of curve editing is to manipulate the tangent handle parameters: Δt^{f}, Δt^{f}, and θ.

BASIC OPTIMIZATION FRAMEWORK WITH EDITABILITY

In this section, we describe the basics of the optimization framework, which is formulated such that the editability is maintained even after optimization. At this moment, we do not consider controllability; we will describe how to add control over optimization in the next section.

Parametrization

Suppose that there are n control handles in the curve editor. In the keyframing stage, animators specify the time t and the value v for each control handle. The remaining degrees of freedom of the i-th control handle, which should be adjusted in the curve editing stage, can be parametrized as

$$\mathbf{x}_i = \begin{bmatrix} \Delta t_i^{\mathrm{b}} & \Delta t_i^{\mathrm{f}} & \theta_i \end{bmatrix}^T \in \mathbb{R}^3, \tag{1}$$

where Δt^{b} and Δt^{f} are the backward and forward displacements of the tangent handle, respectively, and θ is the tangent angle. See Figure 9 for an illustration of the parametrization. By concatenating these vectors over all the target control handles, we can describe the motion as a single vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \cdots & \mathbf{x}_n^T \end{bmatrix}^T \in \mathbb{R}^{3n}. \tag{2}$$

Now, the goal is to adjust the vector $\mathbf{x}$ such that the motion becomes optimal. Let $\mathcal{X} \subset \mathbb{R}^{3n}$ be the domain for this search of the vector $\mathbf{x}$. To ensure that the tangent handles are always valid, the search space $\mathcal{X}$ is defined such that it satisfies the condition: $\Delta t_i^{\mathrm{b}} \leq 0$, $0 \leq \Delta t_i^{\mathrm{f}}$, and $-\frac{\pi}{2} \leq \theta_i \leq \frac{\pi}{2}$ for $i = 1, \ldots, n$.

Objective Function

We formulate the optimization problem as

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} C(\mathbf{x}), \tag{3}$$

where $C(\cdot)$ is a cost function that leads the target motion in a "desirable" direction, and $\mathbf{x}^*$ is an optimal motion obtained through the optimization. The choice of $C(\cdot)$ is arbitrary as long as $C(\cdot)$ appropriately penalizes undesirable motions (*i.e.*, returns a large cost value) and does not penalize desirable motions (*i.e.*, returns a small cost value). In this work, we chose a physically inspired cost function as the first step, whereas other cost functions are possible.

We assume that the cost function can be represented as an integral of an "instantaneous" cost over time. Here, the instantaneousness means that a temporally localized cost value can be evaluated for every moment. From a mathematical viewpoint, this assumption is represented as

$$C(\mathbf{x}) = \int C(\mathbf{x}, t) dt, \tag{4}$$

where $C(\mathbf{x}, t)$ evaluates the cost for the motion $\mathbf{x}$ at the time t. Note that this assumption has been made in many previous successful methods (*e.g.*, [31]).

We implemented a well-established cost function, which was used by Safonova *et al.* [31]. Here is an intuition of this cost function: it penalizes large torques concentrating on specific joints or moments. Mathematically, it is formulated as the sum of squared torques over all the joints:

$$C(\mathbf{x}, t) = \sum_{j \in \mathcal{J}} \|\boldsymbol{\tau}_j(\mathbf{x}, t)\|^2, \tag{5}$$

where $\mathcal{J}$ is the set of all the joints and $\boldsymbol{\tau}_j(\mathbf{x}, t)$ is the torque on the j-th joint for the motion $\mathbf{x}$ at the time t.

The torques are estimated from the motion and the mass properties of the character. This process is called *Inverse Dynamics* [10, 26], and our implementation uses an existing library, RBDL [11]. To calculate the mass properties, we assume that each bone is a rectangular box made of a uniform material.

Solver Details

The analytic derivative of the objective function is not available in our problem setting. Thus, we use BOBYQA, a local derivative-free optimization algorithm [30] implemented in the NLopt library [20]. The motion before optimization is set as the initial solution. Note that we have tested several other algorithms and empirically found that BOBYQA worked well in our setting. We also found that local optimization algorithms are preferable to global ones because global ones are typically slower and often produce motions that are too different from the original motion, which is often undesirable. Moreover, choosing local algorithms is important for the visualization and interaction to be intuitive; most local algorithms update the target parameters gradually, which is easier to understand.

Discussions

In Figure 10, we show a result of applying the optimization to a motion of a stick figure waving his arm. The optimization

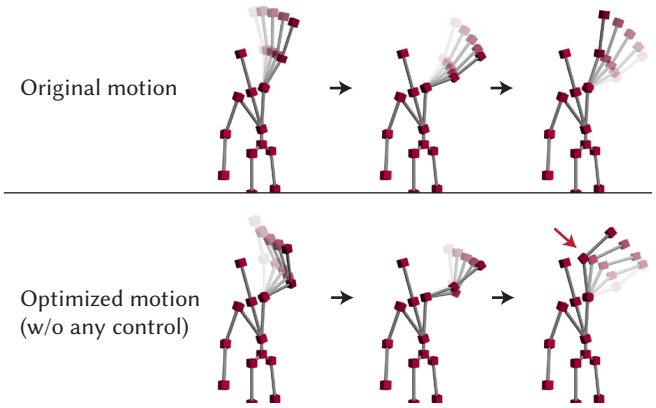

Figure 10. Failure example of optimization *without* any control. (Top) The original motion before optimization. (Bottom) An optimized motion without any control. In this case, the stick figure's elbow is undesirably bent for minimizing its torque. This is natural from the viewpoint of physics, but this is not likely to be the animator's intent.

successfully minimized the torque concentration and provided a physically plausible motion; however, the resulting motion is apparently unnatural because the elbow is bent in a biologically impossible way. This is caused by the fact that the cost function defined by Equation 5 (and probably any other cost function) is not perfect. To overcome this issue and effectively utilize the optimization, control of its behavior is necessary.

ADDING CONTROL OVER OPTIMIZATION

In this section, we present the mathematical formulations for adding controls to the optimization framework described in the previous section.

Control by Regularization

We introduce a regularization term that prevents the resultant motion from being too far from the original motion. By adding this term to the objective function in Equation 3, the original intention is taken into account. Any regularization formulation is acceptable, provided that it penalizes the motion that is far from the original motion; in our implementation, we designed the regularization term D as follows:

$$D = D^{\text{param}} + D^{\text{traj}}, \tag{6}$$

where D^{param} is called *parameter difference* and D^{traj} is called *trajectory difference*.

Parameter Difference

The first term measures the difference between the original motion $\mathbf{x}^{\text{orig}}$ and the resulting motion $\mathbf{x}$ in the parameter space. We formulate it as a weighted squared Euclidean distance between $\mathbf{x}$ and $\mathbf{x}^{\text{orig}}$:

$$D^{\text{param}} = (\mathbf{x} - \mathbf{x}^{\text{orig}})^T \text{diag}(\mathbf{w}^{\text{param}})(\mathbf{x} - \mathbf{x}^{\text{orig}}), \tag{7}$$

where $\text{diag}(\mathbf{w})$ is a diagonal matrix whose i-th diagonal element is w_i, and $\mathbf{w}^{\text{param}} \in \mathbb{R}^{3n}$ is a weight vector for controlling the behavior of regularization, which is specified by the animators indirectly through interface. Each element of $\mathbf{w}^{\text{param}}$ separately controls how much the optimization changes the parameter in the corresponding dimension. If the weight value is set to zero, the optimization freely changes the parameter without considering the original motion. If it is set to an infinitely large value, the optimization does not change the motion.

Trajectory Difference

The second term measures the difference between the trajectories of the joints. We formulate this as

$$D^{\text{traj}} = w^{\text{traj}} \int \left\{ \sum_{j \in \mathcal{J}} \left\| \mathbf{p}_j(\mathbf{x}, t) - \mathbf{p}_j(\mathbf{x}^{\text{orig}}, t) \right\|^2 \right\} dt, \tag{8}$$

where $\mathbf{p}_j(\mathbf{x}, t)$ is a function that returns the position of the j-th joint at the time t when the motion parameter $\mathbf{x}$ is applied, and $w^{\text{traj}} \in \mathbb{R}$ is a weight parameter for controlling the strength of the regularization.

Interface for Specifying Weights

It is technically possible to let animators specify all the values of $\mathbf{w}^{\text{param}}$ and w^{traj} directly; however, this may not be tractable. Therefore, our system uses a single slider, as we showed in the interactive regularization feature, to simultaneously set same values to the weights. In addition to this, to enable the per-joint regularization feature, the system increases the values of the corresponding dimensions of $\mathbf{w}^{\text{param}}$ according to the per-joint slider values specified by the animators.

Control over Cost Function

To let the user control the cost function, we extend the cost function representation in Equation 4 as

$$\hat{C}(\mathbf{x}) = \int w^{\text{cost}}(t) C(\mathbf{x}, t) dt, \tag{9}$$

where $w^{\text{cost}}(t)$ is a time-varying weight function. When $w(t)$ provides a small value for a time t, the cost at the time t is less influential during the optimization, and *vice versa*. The representation of this function can be arbitrary and freely specified by the user. Considering the balance between the expressiveness and the easiness of specification, we chose the following representation: $w^{\text{cost}}(t) = \max\left(1 - \sum_{k \in \mathcal{K}} k(t), 0\right)$, where $k(\cdot)$ is a Gaussian kernel whose center and bandwidth are specified by the animators, and $\mathcal{K}$ is the set of kernels defined by the user. For example, in the case of Figure 8, the number of kernels is one. Note that, if no kernel is specified, the weight function $w^{\text{cost}}(\cdot)$ always returns 1; in that case, $\hat{C}(\cdot)$ in Equation 9 is equivalent to $C(\cdot)$ in Equation 4.

EXAMPLE USAGE SCENARIOS

To demonstrate our approach, we created three example motions using our system (see the video figure for the actual editing processes and motions). Note that the video clips (except for those marked as fast forwarding) were real-time screen captures, which were taken on a MacBook Pro with 3.3 GHz Intel Core i7. As the video shows, the optimization typically converges in a few seconds, and converges within a maximum of around ten seconds.

Stick Figure Waving His Arm

In this scenario, we designed a motion of a humanoid stick figure waving the left arm. The left shoulder, left elbow, right

shoulder, right elbow, and spine were articulated by FK. The total number of the parameters for curve editing was 78.

By applying optimization to all the parameters without any control, we could obtain a physically plausible motion; it modified the spine's motion drastically to decentralize the torque on the left shoulder, and it made the left elbow flexible (see the video figure). However, this motion was undesirable because it looked too different from the original motion, which was not our intention, and moreover, the elbow was bent in an unrealistic way (*c.f.*, Figure 10). To fix these issues, we used the per-joint regularization feature; specifically, we set weights for the spine and elbow joints (*c.f.*, Figure 7). We then re-ran optimization, and obtained a satisfactory motion.

Conductor for a Musical Performance

The goal of this scenario was to create a motion of a cartoon character conducting for a musical performance. The conductor moves his left arm according to the beats of the music. For this scene, we specifically wanted to create sharp movement on the beats and smooth movement for other moments to make the motion look more synchronized with the music. The keyframes were set on the beats and at the middle of the beats. The left arm was controlled by IK, and the other joints were controlled by FK. The total number of parameters was 54.

The initial motion was not very organic (Figure 11; Initial motion). Simply applying the optimization without any control produced a plausible animation, but it was so smooth that we did not feel beats from it (Figure 11; Naïve opt.). To tell the optimizer that we accept large torques on the beat moments, we used the feature of time-varying cost control. Specifically, we set the weight kernels on the beat frames (Figure 11; Weight). With this control, the optimization successfully produced a motion with sharp movements on the beats and physically natural transitions between them (Figure 11; Controlled opt.). Please see the accompanying video figure, in which an effective comparison of these motions is shown with music.

Fox Tail

The goal of this scenario was to create a motion of a fox tail. We performed keyframing so that his tail moves from the rest pose to the right, to the left, and finally to the rest pose again (Figure 12; Initial motion). The tail had three joints at its root, middle, and tip, which were controlled by FK, and the total number of parameters was 27.

First, we applied the optimization without any control. The resulting motion looked physically plausible, but it was so different from the initial motion that we were not satisfied with it (Figure 12; Naïve opt.). Thus, we decided to use the interactive regularization feature; we tweaked the weight parameter for regularization, such that the resulting motion was well balanced in terms of the preservation of the original intent and the physical plausibility (Figure 12; Controlled opt. #1). The obtained motion was satisfactory, but we noticed that this could be improved if the motion from right to left was speedier. To achieve this, we used the time-varying cost control feature; specifically, we put weight kernels between the keyframe where the tail is on the right and the keyframe where it is on the left (Figure 12; Weight). We again performed

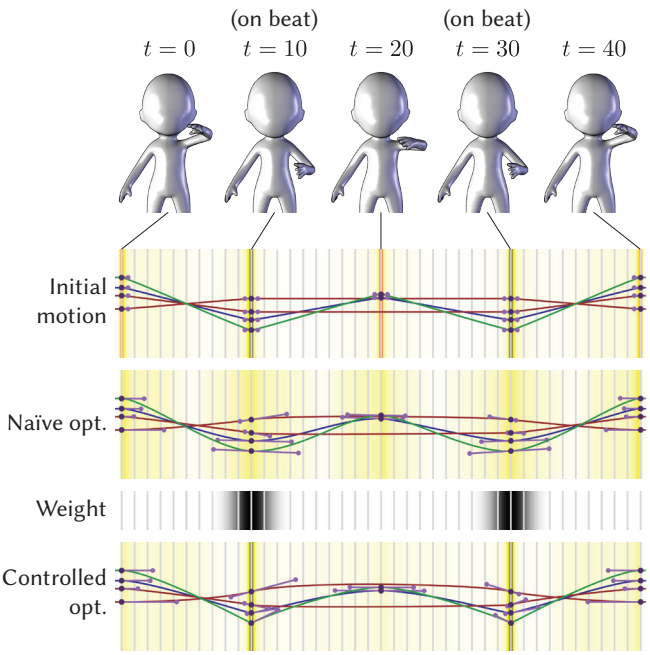

Figure 11. Example scenario: a music conductor. The naïve optimization produced an excessively smooth motion that did not convey the beats well. By using the time-varying cost control feature, the optimization could create sharp movements on the beats.

the optimization with regularization, and finally obtained a desirable motion (Figure 12; Controlled opt. #2). Figure 13 shows a comparison of these motions. It is notable that the *Follow Through* effect [24, 22] (in this case, the tail root leads the motion and the tail tip follows it with a delay) is naturally achieved in the optimized motions.

FEEDBACK FROM EXPERTS

Apart from the demonstration of effective usage scenarios that we showed in the previous section, we also validated our approach through an informal interview with domain experts. Specifically, we interviewed two professional animators (A_1 and A_2), who are familiar with 3D character animation authoring. We interviewed them together. The interview lasted approximately one hour, and it was conducted as follows. First, we asked a question: "*If you can use a new intelligent Maya plug-in that improves curve editing, what characteristics do you expect it to have?*"[2] We then explained our design goals (editability, controllability, and transparency) and asked them about these goals. Next, we explained our system and its features using the scenarios shown in the previous section, and asked them to provide comments for each feature. Finally, we asked them to provide free comments about our system and general character animation. Note that the goal of this interview was not to measure any specific quantities but to obtain qualitative feedbacks on our approach.

Through the interview, we observed that both the interviewees could easily understand our design goals (editability, control-

[2]We conducted the interview in the interviewees' native language, and thus this question and the following comments are the translation from the original language into English.

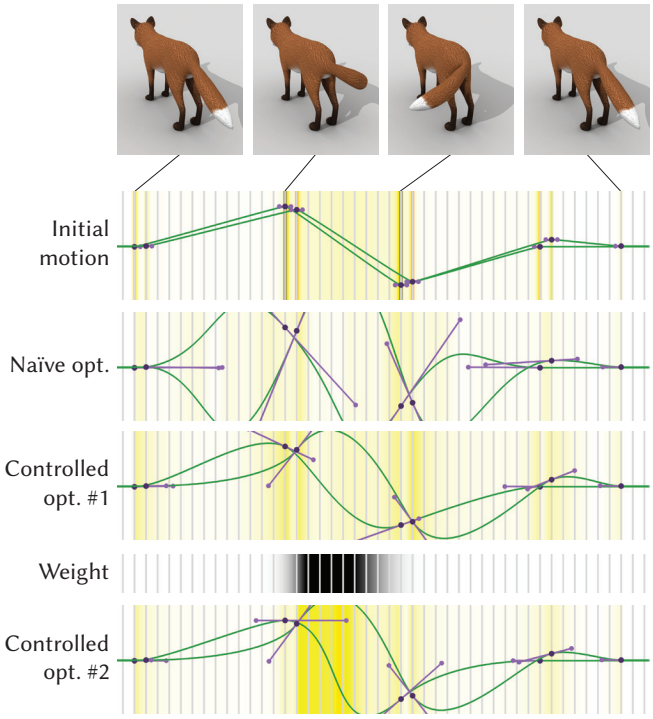

Figure 12. Example scenario: a fox tail. Simply applying the optimization produced a physically plausible but excessively different motion. This was solved by using the interactive regularization feature. Additionally, we used the time-varying cost control feature to speed up the motion from the right to the left. The 3D model is provided by Daniel Moos under CC BY 4.0.

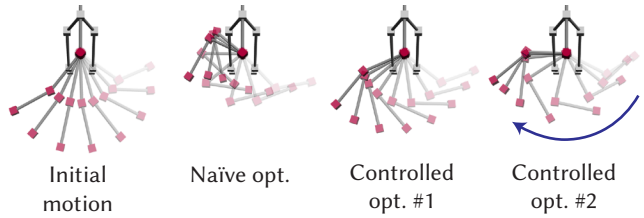

Figure 13. Comparison of the fox tail movements shown as skeletons (top views). The speedy movement in the controlled opt. #2 (indicated by the blue arrow) was achieved by the time-varying cost control feature.

lability, and transparency) and all the features of our system based on a brief introduction.

In response to the first question that we asked before explaining our design goals, A_2 mentioned about the difficulty of achieving physically plausible motions, and she wanted a plug-in that could "*add physical nuance [to animation curves] appropriately.*" A_1 also agreed with the sentiment. Note that we had not told them by this moment about the fact that our system handled physics.

In the following, we summarize the obtained comments sorted by the contents rather than by the chronological order.

Comments on Editability

For the feature of preserving the control handles to ensure editability, A_1 said this feature was "*indispensable*" and also stressed that "*[I] can never imagine the situation where it is desirable that the system automatically adds keys [to animation curves].*" A_2 also agreed with the importance of preserving the handles, and added "*More specifically, it is OK for the [tangent] handles to change, but the keyframes (key points) should not change [through optimization],*" which validates our design.

Comments on Controllability

All the features related to controllability were strongly appreciated by both the interviewees. For example, A_2 said "*I would be puzzled if I can't use this (the per-joint regularization feature).*" When seeing the fox tail animation edited with the time-varying cost control feature (Figure 12; Controlled opt. #2), A_1 said that this feature would be useful "*when I want to add subtle nuance*" and A_2 said she "*often wants to add [such nuance]*" in practical scenarios. For the regularization feature and the time-varying cost control feature, A_1 said that the interactivity of these features could make adjustment "*easier*" and "*less stressful.*"

Comments on Transparency

Both the optimization process and cost visualization features were appealing to the interviewees. For example, A_1 said that the optimization process visualization "*was very interesting*" because she could see how the system performed trials and errors. The interviewees were, however, doubtful as to wether these features could improve "*ease of use*" (A_1) or "*efficiency*" (A_2). An unexpected yet notable point was that these features provided the interviewees with inspiration on how to use and improve our system. Regarding the optimization process visualization, A_2 said she wanted to stop the optimization in the middle of convergence and obtain the intermediate result (rather than just wait for convergence) and suggested the implementation of such a function, while A_1 suggested the implementation of a slider interface to continuously roll back the optimization process. Regarding the cost visualization, A_2 wanted to customize the cost function by herself rather than by using the pre-defined cost function; she raised an example of having a character with a joint that could only move within a limited range of orientations owing to an injury, saying that it would be very nice if she could input such character-specific properties into the cost function definition. Another notable point is that the two interviewees, who were not familiar with either applied math or computer science, could easily understand the concept of numerical optimization by observing the system behavior. We believe this is (at least partially) because the system is transparent.

Comments on Potential Usages in Production

The interviewees mentioned potential uses of our current system in professional production. Whereas both the interviewees agreed that the edited motion of the conductor scene (Figure 11; Controlled opt.) was "*much improved*" (A_1) compared to the original motion (Figure 11; Initial motion), they commented that the quality was still unsatisfactory for a main character in the production. Instead, they commented that it could be used for making "*subcharacters' motions*" (A_2) or in "*projects with only limited time*" (A_1) for saving time. "*It would be great [if I can] use the [saved] time for [other] important characters,*" A_1 added. In contrast to the conductor

scene, they were very excited about the quality of the fox tail animation. Both of them agreed that it was possible to use it even in professional production. A_1 said, "*I think it (the current system) is suitable for less active parts*" such as a fox tail or an arm that does not produce a primary motion. This comment probably stems from the fact that our current system uses a physics-based cost function. As for the *Follow Through* effect in the fox tail motion, A_2 said, "*It is possible to provide it (the* Follow Through *effect) by hand, but it requires much tweaking to make it (the motion) look natural,*" so she appreciated our system. She also commented that it is very tedious to move a girl's skirt because it often has many bones to tweak, so our current system would be useful for such a case as it can easily provide physically plausible motions by only specifying key poses at several keyframes.

Other Comments

A_1 commented that it seemed to be "*easy for novices to create [plausible motions]*" with this system. As for the timing of the optimization usage, A_2 said, "*[I] might use it at the beginning [of the editing]*" rather than the end of editing. A_1 agreed with it, and she added that she would use the result as a starting point for manual refinement. These comments further confirmed the importance of editability. For the cost function, A_2 wanted it to consider project-specific styles of motions; "*Each art project has a specific style of motions,*" and "*it would be nice if it automates [the process of applying such styles]*" by using a customized cost function, A_2 said.

DISCUSSIONS

Summary of Interview

First, animators could understand all the features of our system by a brief introduction. All the features of editability and controllability seem indispensable. The features of transparency are welcomed, but do not seem to improve either ease of use or efficiency; instead, they could provide animators with additional inspiration for uses of our system. The current cost function is not satisfactory and needs improvement to achieve production quality. The current system could be useful in projects with many characters or with limited time, or for motions of non-primary body parts such as fox tails.

Physics Simulation vs. Keyframe Animation

As mentioned in the interview, our system was useful to plausibly create the *Follow Through* effect in some scenarios. Physical simulation could also be used to create similar effects, but it produces only physically "correct" motions and does not provide appropriate editability once the simulation is completed. This makes it difficult or impossible for animators to add artistic intention to motions. In contrast, keyframe animation offers the editability with full and precise control, which helps to achieve the desired motions.

Integration and Evaluation in Professional Workflow

We conducted an interview with domain experts and validated our approach with the proof-of-concept system. The next step would be the integration and testing in professional workflow. Rigorous evaluation in such workflow is necessary. To meet the production-level requirements, we need to address some engineering issues (*e.g.*, support for general rig systems, more appropriate estimation of physical properties, and integration with other existing tools).

Different Cost Functions and Constraints

We chose and implemented a simple physics-based cost function [31] (Equation 5) because we supposed that there was a demand for the support of creating physically plausible motions, which was confirmed in the interview. In the future, we plan to develop various types of cost functions to meet other production needs. For instance, as mentioned in the interview, developing a cost function that considers specific motion styles is important future work; for this, machine learning-based approaches (*e.g.*, [15]) could be used. Considering volumetric deformation [17] may improve the motions of fatter characters, and considering biological properties [36] (*e.g.*, minimization of muscle actuation energy) may be useful to produce biologically natural motions. In addition to investigating other cost functions, it is also interesting when *constraints* are incorporated into the optimization problem; for example, undesirable penetrations could be avoided by adding a constraint that prohibits penetrations in the resulting motions. Note that our framework and interaction designs are independent of the choice of cost functions or constraints.

Support of Keyframing

We focused on the curve editing stage, so the keyframing stage is out of the scope. However, we believe that the use of optimization can enhance the keyframing as well. It is an important future work for investigating system designs to achieve such optimization-guided keyframing.

Other Design Domains

We believe that some of the features we developed for our motion-editing system are general enough and can be useful in other design domains as well. For example, the interactive regularization could be applied to various design frameworks that involve continuous optimization (*e.g.*, photo enhancement) and the visualization features could be implemented in other frameworks (*e.g.*, 2D graphic design). We will explore such potential by investigating domain-specific optimization and visualization methods.

CONCLUSION

We presented a new framework called optimization-guided motion editing, in which numerical optimization is utilized as a tool for animators to effectively edit character motions. To enable this, we set the three design goals: editability of the optimized motion, controllability of the optimization behavior, and transparency of the optimization process. Based on these goals, we presented a set of interactions and their mathematical formulations, and developed a proof-of-concept system, named OptiMo. We showed how this system could support animators by demonstrating several usage scenarios. To validate our approach, we interviewed domain experts using our system, and obtained comments for each feature of the system as well as its potential usages in the professional production.

Acknowledgments

We thank the two interviewees. This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

REFERENCES

1. Adobe Systems Inc. 2017. Adobe After Effects CC. http://www.adobe.com/products/aftereffects.html. (2017).
2. Autodesk Inc. 2017a. Maya. https://www.autodesk.com/products/maya/overview. (2017).
3. Autodesk Inc. 2017b. MotionBuilder. https://www.autodesk.com/products/motionbuilder/overview. (2017).
4. Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4, Article 96 (July 2014), 10 pages. DOI: http://dx.doi.org/10.1145/2601097.2601157
5. Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: Interactive Optimization of Menu Systems. In *Proc. UIST '13*. 331–342. DOI: http://dx.doi.org/10.1145/2501988.2502024
6. Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. 2016. Gesture3D: Posing 3D Characters via Gesture Drawings. *ACM Trans. Graph.* 35, 6, Article 165 (Nov. 2016), 13 pages. DOI: http://dx.doi.org/10.1145/2980179.2980240
7. Samuel R. Buss. 2009. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods. (2009). http://euclid.ucsd.edu/~sbuss/ResearchWeb/ikmethods/index.html
8. Byungkuk Choi, Roger Blanco i Ribera, J. P. Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. 2016. SketchiMo: Sketch-based Motion Editing for Articulated Characters. *ACM Trans. Graph.* 35, 4, Article 146 (July 2016), 12 pages. DOI: http://dx.doi.org/10.1145/2897824.2925970
9. Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sýkora. 2017. Example-based Expressive Animation of 2D Rigid Bodies. *ACM Trans. Graph.* 36, 4, Article 127 (July 2017), 10 pages. DOI: http://dx.doi.org/10.1145/3072959.3073611
10. Roy Featherstone. 2008. *Rigid Body Dynamics Algorithms*. Springer US. DOI: http://dx.doi.org/10.1007/978-1-4899-7560-7
11. Martin L. Felis. 2017. RBDL: an Efficient Rigid-Body Dynamics Library using Recursive Algorithms. *Autonomous Robots* 41, 2 (Feb 2017), 495–511. DOI: http://dx.doi.org/10.1007/s10514-016-9574-0
12. Christoph Gebhardt, Benjamin Hepp, Tobias Nägeli, Stefan Stevšić, and Otmar Hilliges. 2016. Airways: Optimization-Based Planning of Quadrotor Trajectories According to High-Level User Goals. In *Proc. CHI '16*. 2508–2519. DOI: http://dx.doi.org/10.1145/2858036.2858353
13. Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. 2016. Rig Animation with a Tangible and Modular Input Device. *ACM Trans. Graph.* 35, 4, Article 144 (July 2016), 11 pages. DOI: http://dx.doi.org/10.1145/2897824.2925909
14. Michael Gleicher. 1997. Motion Editing with Spacetime Constraints. In *Proc. I3D '97*. 139–148. DOI: http://dx.doi.org/10.1145/253284.253321
15. Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-based Inverse Kinematics. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 522–531. DOI: http://dx.doi.org/10.1145/1015706.1015755
16. Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time Sketching of Character Animation. *ACM Trans. Graph.* 34, 4, Article 118 (July 2015), 10 pages. DOI: http://dx.doi.org/10.1145/2766893
17. Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space Physics. *ACM Trans. Graph.* 31, 4, Article 72 (July 2012), 8 pages. DOI: http://dx.doi.org/10.1145/2185520.2185568
18. Daniel Holden, Jun Saito, and Taku Komura. 2015. Learning an Inverse Rig Mapping for Character Animation. In *Proc. SCA '15*. 165–173. DOI: http://dx.doi.org/10.1145/2786784.2786788
19. Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. 2014. Tangible and Modular Input Device for Character Articulation. *ACM Trans. Graph.* 33, 4, Article 82 (July 2014), 12 pages. DOI: http://dx.doi.org/10.1145/2601097.2601112
20. Steven G. Johnson. 2017. The NLopt Nonlinear-Optimization Package. (2017). http://ab-initio.mit.edu/nlopt
21. Andreas Karrenbauer and Antti Oulasvirta. 2014. Improvements to Keyboard Optimization with Integer Programming. In *Proc. UIST '14*. 621–626. DOI: http://dx.doi.org/10.1145/2642918.2647382
22. Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion Amplifiers: Sketching Dynamic Illustrations Using the Principles of 2D Animation. In *Proc. CHI '16*. 4599–4609. DOI: http://dx.doi.org/10.1145/2858036.2858386
23. Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-powered Parameter Analysis for Visual Design Exploration. In *Proc. UIST '14*. 65–74. DOI: http://dx.doi.org/10.1145/2642918.2647386
24. John Lasseter. 1987. Principles of Traditional Animation Applied to 3D Computer Animation. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 35–44. DOI: http://dx.doi.org/10.1145/37402.37407

25. J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. In *Eurographics 2014 – State of the Art Reports*. DOI: http://dx.doi.org/10.2312/egst.20141042

26. Xiaolei Lv, Jinxiang Chai, and Shihong Xia. 2016. Data-driven Inverse Dynamics for Human Motion. *ACM Trans. Graph.* 35, 6, Article 163 (Nov. 2016), 12 pages. DOI:http://dx.doi.org/10.1145/2980179.2982440

27. J. McCann, N. S. Pollard, and S. Srinivasa. 2006. Physics-based Motion Retiming. In *Proc. SCA '06*. 205–214. http://dl.acm.org/citation.cfm?id=1218064.1218092

28. Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proc. CHI '15*. 1221–1224. DOI: http://dx.doi.org/10.1145/2702123.2702149

29. A. Cengiz Öztireli, Ilya Baran, Tiberiu Popa, Boris Dalstein, Robert W. Sumner, and Markus Gross. 2013. Differential Blending for Expressive Sketch-based Posing. In *Proc. SCA '13*. 155–164. DOI: http://dx.doi.org/10.1145/2485895.2485916

30. Michael J. D. Powell. 2009. *The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives*. Technical Report NA2009/06. Department of Applied Mathematics and Theoretical Physics, Cambridge England.

31. Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. 2004. Synthesizing Physically Realistic Human Motion in Low-dimensional, Behavior-specific Spaces. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 514–521. DOI: http://dx.doi.org/10.1145/1015706.1015754

32. Ari Shapiro and Sung-Hee Lee. 2010. Practical Character Physics for Animators. *IEEE Computer Graphics and Applications* 31 (2010), 45–55. DOI: http://dx.doi.org/10.1109/MCG.2010.22

33. Kwang Won Sok, Katsu Yamane, Jehee Lee, and Jessica Hodgins. 2010. Editing Dynamic Human Motions via Momentum and Force. In *Proc. SCA '10*. 11–20. DOI: http://dx.doi.org/10.2312/SCA/SCA10/011-019

34. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proc. DIS '16*. 543–555. DOI: http://dx.doi.org/10.1145/2901790.2901817

35. Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes. *ACM Trans. Graph.* 33, 4, Article 65 (July 2014), 10 pages. DOI:http://dx.doi.org/10.1145/2601097.2601129

36. Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (July 2012), 11 pages. DOI: http://dx.doi.org/10.1145/2185520.2185521

37. Andrew Witkin and Michael Kass. 1988. Spacetime Constraints. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 159–168. DOI: http://dx.doi.org/10.1145/378456.378507