SketchiMo: Sketch-based Motion Editing for Articulated Characters

Byungkuk Choi[1*] Roger Blanco i Ribera[1*] J. P. Lewis[2] Yeongho Seol[2] Seokpyo Hong[1]
Haegwang Eom[1] Sunjin Jung[1] Junyong Noh[1]

[1]KAIST [2]Weta Digital

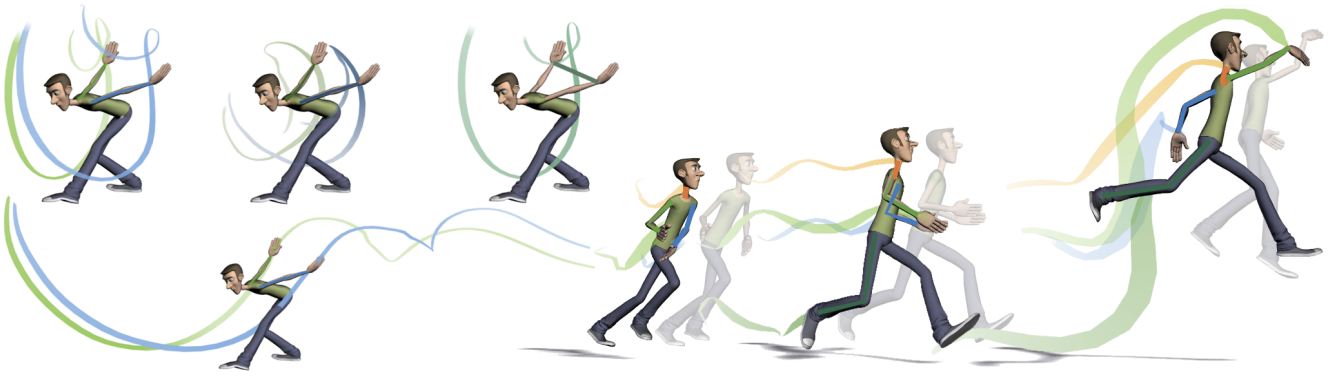

Figure 1: *Left: SketchiMo offers different visualizations that accentuate different aspects of a motion: a joint path in world space (top left), the relationship between a joint and its parent (top middle), between two coordinated body parts (top right), or the temporal character of the movement (bottom). All the highlighted lines are editable via sketch input. Right: An edited dunk motion performed with SketchiMo compared with the original motion. (Malcolm character courtesy of AnimSchool.com)*

Abstract

We present SketchiMo, a novel approach for the expressive editing of articulated character motion. SketchiMo solves for the motion given a set of projective constraints that relate the sketch inputs to the unknown $3D$ poses. We introduce the concept of sketch space, a contextual geometric representation of sketch targets —motion properties that are editable via sketch input— that enhances, right on the viewport, different aspects of the motion. The combination of the proposed sketch targets and space allows for seamless editing of a wide range of properties, from simple joint trajectories to local parent-child spatiotemporal relationships and more abstract properties such as coordinated motions. This is made possible by interpreting the user's input through a new sketch-based optimization engine in a uniform way. In addition, our view-dependent sketch space also serves the purpose of disambiguating the user inputs by visualizing their range of effect and transparently defining the necessary constraints to set the temporal boundaries for the optimization.

Keywords: sketch-based interface, motion editing, character animation, articulated character motion

Concepts: •**Human-centered computing** → **Graphical user interfaces;** •**Computing methodologies** → **Motion processing;**

*Both authors contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. © 2016 ACM.
SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925970

1 Introduction

Most visual artists rely on sketching, in one way or another, to develop their ideas. Sketching is used in all phases of the creative process, from building up shapes and exploring motion with rough key poses to drawing storyboards. Sketch-based interfaces have been widely used for various computer graphics applications because they allow for expressive, yet simple and intuitive, interaction, in a way that is closer to cognitive processes [Annett et al. 2014]. Researchers in computer graphics have been keen on drawing inspiration from more traditional creative processes. Major developments in sketch-based research range from modeling and rigging [Igarashi et al. 1999; Borosán et al. 2012; Levi and Gotsman 2013; De Paoli and Singh 2015], sculpting and mesh editing [Nealen et al. 2005; Kho and Garland 2005; Takayama et al. 2013], to $3D$ painting [Schmid et al. 2011] even to the simulation of rigid body objects [Popović et al. 2003; Thorne et al. 2004].

Sketch-based interfaces have found their way into production, a milieu dominated by the well-established WIMP[1] paradigm and with a certain tendency to resist the introduction of dramatic shifts into long-held production practices. Existing WIMP interfaces include a number of different editors such as the graph, outliner, and channel editors in Maya or MotionBuilder [Autodesk a]. These provide exact, abstract, and complex control, but at the expense of a steep learning curve. In addition, these editors separate the animation process from the actual character, and a large portion of the animation process is spent simply switching between different editing windows.

In contrast, sketch-based interfaces exploit the direct relationship between the sketched input and the resulting effect in a WYSIWYG[2] way. With the proven success of specialized sketch-based software for modeling [Pixologic ; Autodesk b], sketch-based interfaces have slowly seeped into more traditional CG software such

[1]window, icon, menu, pointer
[2]what you see is what you get

as Maya and Blender [Blender Foundation], through their sculpting and texturing tools among others. On the flip side, sketch-based interfaces also introduce their own challenges. Considering that articulated figures have relatively high degrees of freedom (DoFs) to control, and the difficulty of editing $3D$ elements with inputs on a $2D$ space, it is not surprising that fully sketch-based animation of articulated characters remains a challenging and open problem.

Rather than creating an animation from scratch, existing captured motion could be edited to impose a desired style or satisfy other constraints. However, motion capture intrinsically lacks keyframes or other high-level controls that would support such editing. As a result, it is often easier to recapture a new motion than it is to alter existing motion. Research has mainly focused on synthesizing new motion from available motion databases [Yoo et al. 2014] and high-level indirect manipulation of dense motion data through sparse spacetime constraints [Gleicher 1997; Gleicher 1998; Lee and Shin 1999; Gleicher 2001; Kim et al. 2009; Ho et al. 2010]. All in all, the lack of efficient and intuitive tools to directly manipulate dense motion data has prevented motion capture from establishing a foothold in stylistic animation pipelines, where keyframing and rig manipulations remain the favored approach.

We propose SketchiMo, a sketching interface for the expressive editing of articulated character motion. We solve this highly underconstrained problem through the combination of two user interface constructs, *sketch targets* and *sketch spaces*, together with an underlying optimization engine. Sketch targets (see Sec. 4) are objects such as body lines or trajectories that are editable via sketch input. Sketch spaces (see Sec. 5) are dynamic, view-dependent pictorial representations of the animation that accentuate, right on the viewport, different aspects of the motion. In the *global* sketch space the animator sees snapshots of the character at different times (Fig. 1, top left), and can alter either body lines or temporal joint path targets by sketching the new desired curves. In the *local* sketch space the animator can observe and edit the trajectory of a joint relative to its parent (Fig. 1, top middle). In the *dynamic* sketch space (Fig. 1, bottom left), time is dilated, allowing fine control over the temporal aspects of the motion.

Our novel optimization solves for the new $3D$ character motion that is consistent with the projective constraints provided by the artist's sketched curves. The interface is both simple and fluid, yet when combined with the underlying optimization it is also powerful —our system allows a variety of goals to be seamlessly handled, ranging from forward kinematics (FK) and inverse kinematics (IK) to abstract properties such as average paths and and inter-part constraints. The resulting motions can also be iteratively refined during the sketching process.

Contributions SketchiMo extends previous research on sketch-based editing [Guay et al. 2013; Guay et al. 2015b] to provide, for the first time, entirely sketch-based motion editing of articulated characters. All editing occurs *directly on the viewport*, using nothing more than sketched strokes, with no need for auxiliary editors (e.g. graph, channel, attribute editors). *Direct and fine control* is provided: the artist directly sketches the desired shape of the motion or pose, rather than working with sparse control handles. *Multiple aspects* of the motion can be edited, including pose, trajectories, timing, and relationships, using the same interface. Although the resulting interface is particularly simple and intuitive, achieving this simplicity required solving problems resulting from the ambiguities of a $2D$ viewport and the many DoFs of a humanoid character. In particular we solve the problem of selecting the intended target of an edit, and show that the sketch space interface construct can reliably disambiguate the user's inputs by visualizing their range of effect and transparently defining the temporal boundaries for the optimization.

2 Related Work

Early works in spacetime motion editing developed motion warping techniques allowing the user to smoothly modify given motions while satisfying a set of specified constraints [Gleicher 1998; Lee and Shin 1999]. Gleicher [2001] abstracted the global motion of a character into a $2D$ motion path which in turn could be directly manipulated through handles. Recently, higher level editing scenarios such as multiple character synchronization [Kim et al. 2009] or retargeting with spatial relationship considerations [Ho et al. 2010] have been demonstrated. In these studies, only a sparse set of constraints is needed in order to achieve the desired high-level manipulations of the given data. More detailed modifications, such as stylistic editing of the motion, would require a sensibly larger number of constraints. We manage dense constraints for fine motion editing from input sketches with the selected sketch targets and sketch space in a unified optimization framework.

Motion stylization Stylistic editing and synthesis of motion have been approached by learning motion patterns from captured example sequences [Brand and Hertzmann 2000; Grochow et al. 2004], that have equivalent motions with different styles [Hsu et al. 2005], or by decomposing motion into visually meaningful style components [Shapiro et al. 2006]. Wang et al. [2006] designed a motion filter based on the traditional animation techniques [Lasseter 1987]. By explicitly defining style, these methods restrict editing to their specific definition of style. Instead, our work is inspired by Guay et al. [2013] who provided an intuitive tool capable of achieving stylistic expression, but whether the resulting edit reflects a confident stance or a laid back attitude is left up to the artist's discretion.

Pose and timing manipulation Previous studies aiming for a more direct motion control suggested separate pose and timing editing methods. For character posing, skeleton-based parameterizations are widespread in both graphics research and production. $2D$ stick figures have been used to specify poses [Lin et al. 2010; Wei and Chai 2011; Choi et al. 2012]. A further abstraction of the character skeleton, the line of action, proved capable of simple yet expressive pose manipulations [Guay et al. 2013; Öztireli et al. 2013]. Hahn et al. [2015] generalized the abstraction by letting the user define the actual sketch-based abstraction.

Time warping techniques [Witkin and Popovic 1995; Rose et al. 1998; Hsu et al. 2007] allow for timing modification without affecting poses to adapt the motion to meet new timing constraints. Efforts to improve the manipulation of timing have included letting the user "act" it directly by gesturing [Terra and Metoyer 2004], augmenting the timeline with editable pose-icons and drag-and-drop operations [Mukai and Kuriyama 2009], or visualizing timing directly on the viewport with a set of deformable spatial planes along the motion [Yoo et al. 2015]. We visualize timing through our sketch space: The thickness of lines and the dynamic space help visualizing speed and timing.

Expressive motion editing Similar to our goal, a few studies have focused on the editing of expressive aspects of character motion. Neff and Fiume [2003] developed a set of tools for adjusting succession, amplitude and extent of the motion. Coleman et al. [2008] introduced the concept of staggered poses, allowing densely-sampled motion to be represented by locating coordinated movement features and modeling motion detail using splines and displacement maps. Guay et al. [2015a] extended their work on the line of action [Guay et al. 2013] by adding dynamics in order to interpolate between successive lines of actions. Spacetime sketching of character animation [Guay et al. 2015b] is perhaps the closest work to ours. They bridged the gap between pose and time editing by creating character motion from a single user sketch. Further refinement could be achieved with additional input. However, their

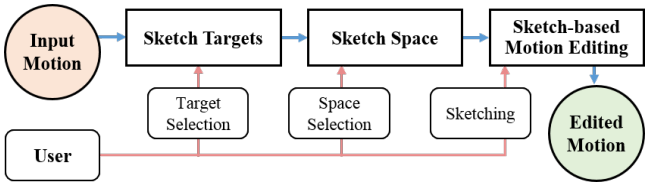

Figure 2: *Overview: In a typical editing iteration in SketchiMo, the user selects a target and a space, then proceeds to edit the motion data through sketching.*

technique focused on relatively simple characters without limbs. We build upon the direction established by [Guay et al. 2015b] by developing a small set of spacetime sketch abstractions that allow natural stroke based editing of complex moving biped characters.

Keyframing Finally, despite being generally considered the standard approach in the industry, the keyframe approach to animation is not without faults. There is a clear separation between spatial and temporal manipulations which not only hinders the coordination of pose evolution over time but also requires switching between tools, viewports and editors. Pose manipulations are mainly performed in the viewport by manipulating a control rig structure, skillfully crafted by character TDs and mostly based on FK and IK, different types of constraints, or set-driven key among others [McLaughlin et al. 2011]. The usability and functionality of a rig is highly dependent not only on the skill and time invested in the rig creation but also on the animation needs. For timing, the animator is mainly directed towards the graph editor [Autodesk a; Coleman et al. 2008] where the temporal curves for different rig parameters are presented. Since these curves represent rig parameters, there is a dissociation between what happens in the graph editor and the viewport, i.e., in an arm FK type of rig, the trajectory of the wrist is the accumulated rotations of all the joints in the chain. This results in a back-and-forth switching between editors in order to achieve the desired results. In contrast, our approach could be considered as a dynamic rigging approach, where the sketch targets and sketch space enable a simple and fast selection process. The user can visualize and edit different aspects of both pose and motion directly on the viewport.

3 Overview

This work introduces SketchiMo, a framework that achieves motion editing for articulated characters through a sketching-only interface. The key challenge underlying SketchiMo, that is, controlling a complex character with multiple DoFs using only $2D$ strokes, is inherently a highly underdetermined problem. To address this challenge, we devised a set of novel interface constructs, sketch targets and sketch spaces, where various editing intents can be naturally accommodated, together with a sketch-based IK optimization that combines the available information and constraints to produce the desired $3D$ motion edit (Fig. 2).

Specifically, we provide a flexible target selection method to specify a region of interest on the skeleton of the character. Through our selection method, the user is presented with a set of sketch targets ready to be edited via input strokes. Our selection method is easy to use and greatly reduces ambiguities in a consistent way (Sec. 4).

Depending on the editing purpose, the user can select a sketch space which provides different contextual visualizations of the selected sketch targets. We provide three —global, local, and dynamic— sketch spaces inspired by the animator's workflow. Global and local spaces are motivated by IK, FK manipulations, respectively, and the dynamic space by the need to temporally disambiguate motions that might spatially overlap (Sec. 5).

As a backend, our system adopts a hierarchical motion fitting technique with a new sketch-based IK solver. During the sketching process, a single sketch target is determined automatically. Projective constraints relating the sketch and the unknown $3D$ poses are then built in accord with the target type and chosen space. The sketch-based IK formulation integrates the different types of constraints into the optimization in a uniform way (Sec. 6).

We demonstrate that our method can be easily extended with additional types of targets to address more specific editing scenarios such as editing coordinated motion (*average path*), motion synchronization (*relational line*), or noise removal by brushing on the path during the sketching process (Sec. 7).

4 Sketch Targets

Animation is often approached in a coarse-to-fine way. From an initial rough motion, the animation is slowly crafted with incrementally finer edits. There are different needs in the editing of the motion at different iterations, which in turn are also influenced by the specific type of animation in hand. Well designed rigs have a priori redundant functionality, such as FK and IK controllers acting on a same limb or different parenting switches for the IK end effector, in order to ease the keyframing process for different animation scenarios. These overlapping controllers address radically different editing needs. For instance, free movements such as dance hand gestures might be simpler to animate with FK controllers while catching a ball with the hand might be easier with IK controllers.

We approach this challenge from a dynamic rigging point of view in the sense that by providing a flexible selection of editable motion properties, the user can locally and temporally rig the character depending on the intended editing. By flexibly selecting sketch targets, we provide control over what is being edited within the character's body and clearly delimit the effective range of the IK behavior when solving the motion with a simple dragging motion.

In the following sections we first mathematically define the nature of each sketch target and then describe the selection process.

4.1 Sketch Target Representation

A *sketch target* can be thought of as a line in the scene ready to be sketched by a user. Formally, a sketch target is represented as a $3D$ polyline $S = \{\mathbf{s}_0, \ldots, \mathbf{s}_{N-1}\}$ composed of N connected vertices. We define two primary sketch targets —*body lines* and *joint paths*— that can be extracted from a given motion.

A **Body Line** is defined as a sub-chain of joints in the character skeleton structure, that is, a connected $1D$ joint chain that contains a set of joint positions at time t

$$S^{bl} = \left\{ \mathbf{s}_j^{bl}(t) \in \mathbb{R}^3 \;\middle|\; \forall j \in J^{chain} \right\}. \tag{1}$$

Multiple body lines can be manually defined on arbitrary parts of the body. Similar to recent approaches [Öztireli et al. 2013; Guay et al. 2013; Guay et al. 2015b], our definition of the body line is concise, but provides the user with freedom to sketch specific body parts by manually selecting them.

A **Joint Path** is a set of ordered positions

$$S^{jp} = \left\{ \mathbf{s}_j^{jp}(t) \in \mathbb{R}^3 \;\middle|\; 0 \leqslant t < N_t \right\}, \tag{2}$$

which are uniformly sampled from the trajectory of joint j across time t. N_t represents the total number of frames to be edited. The joint path gives the user the ability to add expressive variations within a specific time interval. This range is modified as the user changes the current time or camera view. The polyline is rendered with a varying thickness matching the magnitude of the velocity of the joint. This visual feedback can be exploited to edit certain timing aspects such as the ease-in or ease-out of the motion (see Sec. 7.2).

Without loss of generality, we will notate a set of sketch targets as follows:

$$\mathbb{S} = \{S_i \mid 0 \leqslant i < N_s\}, \tag{3}$$

where N_s is the total number of selected sketch targets. Note that S_i can be either S^{bl} or S^{jp}. We also denote $\mathbf{s}_j^i(t)$ as a general target position, specified in the ith sketch target on joint j at time t. The exact positions of a sketch target depend on the current choice of sketch space which will be explained in Sec. 5.

4.2 Sketch Target Selection

Selecting the proper sketch targets is an important step as they reflect user's editing intention, and may vary depending on a given character and its motion. We design an intuitive and easy-to-use selection method relying on the following criteria:

- **Flexibility** Any arbitrary body line (joint sub-chain) from an arbitrary character that has a tree-structured skeleton configuration can be selected as a sketch target.
- **Usability** User interaction must be suitable for pen inputs. We consider a dragging motion rather than a picking motion. The system determines the closest joints and visualizes the current selection in real-time.
- **Consistency** All selections are made by specifying two end joints with a dragging motion. Every selection produces a body line and its associated joint path targets.

To satisfy all these criteria, we first let the user select two joints with a dragging motion. The shortest path in the skeleton hierarchy between both joints is then used as a single body line target. Joint path targets are selected by considering the two extreme joints on the hierarchy of the selected body line. Specifically, we produce a joint path only if an extreme joint is a leaf joint. This means that, depending on the selected body line, at most two joint path targets will be offered to the user for editing (Fig. 3).

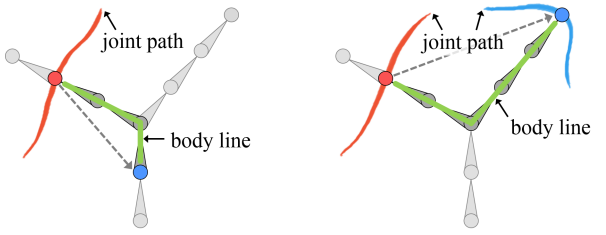

Figure 3: *Sketch target selection examples. Left: A single body line and a joint path. Right: A single body line and two joint paths, note the red and the blue joints are both leaf joints in the selected joint chain, leading to the creation of two joint paths.*

Finally, we put a positional constraint on the joint in the top-most hierarchy of the selected body line. With this, sketching will only affect the region highlighted by the selected sketch targets. By selecting a single joint and its immediate parent, an FK type of control can be attained, whereas dragging the selection further down the line of the hierarchy allows IK behavior.

Figure 4: *Global and local spaces. Top: Motion of the left wrist in the global space (left) and in the local space w.r.t. the shoulder (right). The local space reveals a simple backward swing of the arm followed by a larger forward swing. In this case, the local space permits a more intelligible display of the extent and orientation of the arching motion. Bottom: An edited motion (left) with a single stroke on the local space (right).*

5 Sketch Space

In this section, we define a core concept, the *sketch space* —a geometric representation of sketch targets in $3D$ space. This space is dynamic and view-dependent in the sense that it is designed to be switched or transformed, and automatically delimits the temporal editing bounds as the user navigates the $3D$ viewport.

5.1 Sketch Space Transformation

In order to suit the user's specific editing intentions effectively, the transformed spaces serve to enhance or visualize directly on the viewport different aspects of the motion that might be otherwise hidden with the traditional representation of joint paths. We propose three different spaces: global, local, and dynamic.

Global space The global space is the default sketch space. It corresponds to the $3D$ world space of the application. In the global space, we represent the evolution of the different targets in time and space. In this sense, sketch targets in the global space coincide exactly with the motion. Because of this intuitiveness, the use of global space has been the standard representation in conventional motion editing process.

Specifically, let $\mathcal{M}(t) = (\mathbf{p}_0(t), \mathbf{q}_0(t), \ldots, \mathbf{q}_{N_j-1}(t))$ denote a given motion, where $\mathbf{p}_0(t) \in \mathbb{R}^3$ and $\mathbf{q}_i(t) \in \mathbb{S}^3, (0 \leqslant i < N_j)$ describe the translational motion of the root segment and the rotational motion of the ith joint at time t, respectively, and N_j is the number of joints. Then, we can compute a global point on a sketch target from a joint j at time t as follows:

$$\mathbf{s}_j(t) = \Big\{ \Big(\mathbf{p}_0^b + \mathbf{p}_0(t), \mathbf{q}_0^b \mathbf{q}_0(t)\Big) \otimes \Big(\mathbf{p}_1^b, \mathbf{q}_1^b \mathbf{q}_1(t)\Big) \otimes \cdots \otimes \Big(\mathbf{p}_{j-1}^b, \mathbf{q}_{j-1}^b \mathbf{q}_{j-1}(t)\Big) \Big\} \mathbf{p}_j^b. \tag{4}$$

Here, $(\mathbf{p}_A, \mathbf{q}_A) \otimes (\mathbf{p}_B, \mathbf{q}_B) = (\mathbf{q}_A \mathbf{p}_B \mathbf{q}_A^{-1} + \mathbf{p}_A, \mathbf{q}_A \mathbf{q}_B)$ [Lee 2008], and the superscript b denotes the value from the initial body

configuration which is constant. For notational simplicity, we assume that joint indices in a chain from a selected joint j to the root joint are arranged in a descending order.

Local space The trajectory of a joint in the global space is the cumulative composition of the different motions of the joints down the line of the character's skeletal hierarchy. While in general it is very intuitive to manipulate the motion in this way, it is sometimes necessary to edit it with respect to a certain part of the body (Fig. 4).

Our target selection allows to define arbitrary parental relations. Local space changes the visualization by fixing the parent transformation across time, with the result that in the effect of the joints higher in the hierarchy is removed from the final visualization.

We define $\mathbf{s}_{j\to k}(t)$ as a point in the local sketch space such that the point lies in a local path of the selected joint j with respect to the frame joint k at time t. Specifically,

$$\mathbf{s}_{j\to k}(t) = \mathbf{M}_k(t_c)\left\{\left(\mathbf{p}^b_k, \mathbf{q}^b_k\mathbf{q}_k(t)\right)\otimes\cdots\otimes\left(\mathbf{p}^b_{j-1}, \mathbf{q}^b_{j-1}\mathbf{q}_{j-1}(t)\right)\right\}\mathbf{p}^b_j, \tag{5}$$

where t_c is the current time chosen by a user, and

$$\mathbf{M}_k(t_c) = \left(\mathbf{p}^b_0 + \mathbf{p}(t_c), \mathbf{q}^b_0\mathbf{q}_0(t_c)\right)\otimes\cdots\otimes\left(\mathbf{p}^b_{k-1}, \mathbf{q}^b_{k-1}\mathbf{q}_{k-1}(t_c)\right) \tag{6}$$

is the fixed transformation matrix of joint k with respect to the root joint at time t_c. Note that a fixed transformation matrix $M_k(t_c)$ does not change over time t while computing sketch target points.

Dynamic space Motion is represented in the screen as a $2D$ slice of $4D$ data, that is, $3D$ poses evolving in time. Changing the point of view alleviates the lack of a spatial dimension, but cannot cope with the visualization of time. The representation of motion through body lines or trajectories will suffer from spatial superposition of poses within a certain time interval, complicating the editing of the motion. In the global space, for motions with large displacements such as running, the superposition may not be a big problem but it would certainly arise for slow or close to stationary motions. Furthermore, with the introduction of the local space, there are increasing chances that a body part appears static with respect to the defined parent.

To address this we warp the sketch targets by adding a previously designed warping vector as follows:

$$\mathbf{s}_j^{warped}(t) = \mathbf{s}_j(t) + \alpha\mathbf{w}(t), \tag{7}$$

where α is a user parameter that determines a degree of warping.

Considering the selected sketch target and the contextual state of the motion, we can design warping vectors in order to explicitly visualize time on the joint path (Fig. 5). We utilize the axis orthogonal to the viewing direction $\mathbf{x}_{camera}$ to evenly distribute the frames so that the visualized spatial gap between frames corresponds proportionally to the temporal gap between those same frames:

$$\mathbf{w}(t) = \underbrace{-\left((\mathbf{s}_0(t) - \mathbf{s}_0(t_c))\cdot\mathbf{n}\right)\mathbf{n}}_{\text{warp towards a virtual plane}} + \underbrace{(t - t_c)\,\mathbf{x}_{camera}}_{\text{warp along horizontal axis}}, \tag{8}$$

where $\mathbf{n}$ is the normal vector of the plane orthogonal to the viewing direction (Fig. 6).

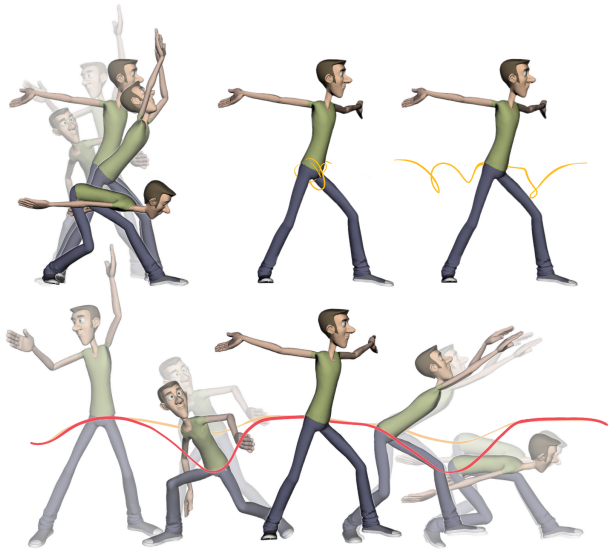

Figure 5: *Dynamic space. Top: Motion with a significant pose overlap (left). The root joint path in the global space (middle). Increasing the warping factor declutters the trajectory (right). Bottom: The warping factor can be increased until the overlap is completely removed (yellow) in order to simplify the editing process (red).*

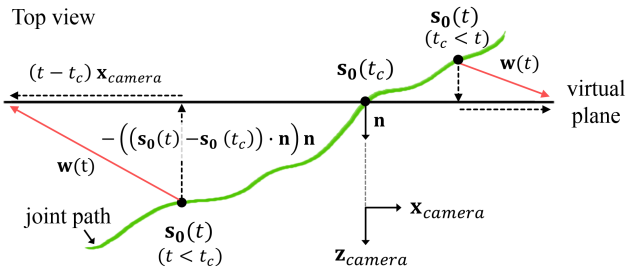

Figure 6: *Dynamic warping vector.*

5.2 View-dependent Selection of Editing Boundaries

Another purpose of sketch space is to disambiguate the sketch inputs by visualizing the range of effects and setting the constraints for the optimization, transparently to the user. Our design choices mainly stem from observation of the workflow of a professional animator. In the traditional process of animating, the animator often begins by choosing a central keyframe, an important instant in the motion where the artist might want to add more dramatic pose variations. Next, they fix a viewpoint and delimit a certain time domain to edit.

We observed that these choices of the time range and the camera viewpoint are influenced by the editing intention. The choice of viewpoint helps the artist focus on the editing in the time range where the motion is visible within screen space. For example, a close-up view on a specific body part reflects an intention to refine motion details. Navigating away reveals global behaviors of the character. Because the screen shows a $2D$ projection of the actual $3D$ space, the choice of viewpoint signals the intention of modifying motion properties that are generally tangent to the viewing plane. Previous research has exploited analogous assumptions. For example, sketch-based modeling paradigms often assume that the user will rotate the model so that the silhouette best reflects

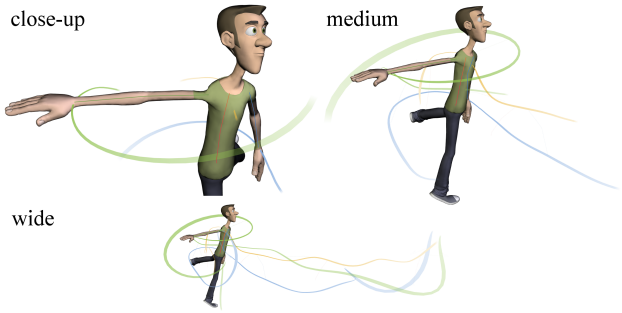

Figure 7: *View-dependent editing boundary. As the user navigates far from the character the editable motion range increases.*

the region they intend to edit [Igarashi et al. 1999; Nealen et al. 2005]. Thus, given a central current frame, the sketch space is automatically equipped with a set of proper constraints for the selected sketch targets as the user navigates the viewport.

We choose the editing range for the optimization by considering the distance between the character and the camera (Fig. 7). The rationale behind this criterion is that a small number of frames are selected when a viewpoint is close to the target path, while more frames are added as the camera moves away. Specifically, a set of bounded time frames T^b in a specific camera view at the current time t_c is chosen as follows:

$$T^b = T^- \cup T^+ = \{t_c - m, \ldots, t_c, \ldots, t_c + n\}, \qquad (9)$$

where

$$T^- = \left\{ t_c - m, \ldots, t_c \;\middle|\; m < f\left(\bar{\mathbf{s}}(t_c)\right), \sum_{t=t_c-m}^{t_c} l(t) < \eta \right\},$$
$$T^+ = \left\{ t_c, \ldots, t_c + n \;\middle|\; n < f\left(\bar{\mathbf{s}}(t_c)\right), \sum_{t=t_c}^{t_c+n} l(t) < \eta \right\}. \qquad (10)$$

Here, m and n represent the number of preceding and succeeding frames, respectively, $\bar{\mathbf{s}}(t) \in \mathbb{R}^3$ is the centroid of a character pose at time t, and $f(\mathbf{x})$ computes the number of bounding frames proportional to the depth of a point $\mathbf{x}$. $l(t) = \|\text{proj}(\bar{\mathbf{s}}(t) - \bar{\mathbf{s}}(t-1))\|$ is the length between two consecutive centroids in screen space and is used to clip the sketch space within a user threshold value (2η). Note that the editing boundary changes upon camera motion, as $f(\mathbf{x})$ and $l(t)$ are view-dependent.

6 Sketch-based Motion Editing

We next discuss how a given motion can be edited from sketches. After a user selects sketch targets, an editing time t_c and the current camera view, the $2D$ stroke the user draws is used to constrain one of the sketch targets. We then fit a given motion to the sketch in the bounded time region. The edited motion is immediately shown to the user, and then refined by simply adding more strokes.

Our objective is to position the selected sketch target close to the sketched stroke while preserving the original motion as much as possible. Accomplishing this requires first selecting the best matched sketch target among multiple targets, and then optimizing the motion with a set of constraints. As our objective incorporates optimization of the motion within different types of sketch space, we design a constraint function that can cope with each space in a uniform way.

6.1 Selecting a Single Sketch Target

As multiple targets are usually present in the sketch space, the specific target being sketched by the user should be first determined. We consider which sketch targets are most visible in the current view. This can be done automatically whenever the user changes the camera view by taking advantage of the sketch space described in Sec. 5. Consequently, the user can simply sketch a stroke over the desired target out of multiple targets without needing to separately select it. The sketch target is selected by minimizing

$$S^* = \arg\min_{S_i \in \mathbb{S}} E(S_i, C^s), \qquad (11)$$

where

$$E(S_i, C^s) = \min(\|\bar{S}_i - \bar{C}^s\|, \|\bar{S}_i - \bar{C}^s_{reverse}\|),$$
$$\bar{S}_i = \begin{bmatrix} \text{proj}(S_i)(p_0) \\ \vdots \\ \text{proj}(S_i)(p_{r-1}) \end{bmatrix}, \quad \bar{C}^s = \begin{bmatrix} C^s(p_0) \\ \vdots \\ C^s(p_{r-1}) \end{bmatrix}.$$

$\bar{S}_i$ and $\bar{C}^s$ are the column vectors in $\mathbb{R}^{r\times 2}$ concatenating r representative points of the ith projected sketch target and a user sketch stroke, respectively. We simply use five evenly spaced samples on $\bar{S}_i$ and $\bar{C}^s$ after parameterizing both over $[0, 1]$ $(p_0 = 0.0, \ldots, p_{r-1} = 1.0)$. $\bar{C}^s{}_{reverse}$ is the vector containing the same points as $\bar{C}^s$ in a reverse order. We take the minimum value after comparing both directions of the sketch stroke to the sketch targets to account for the freedom of sketching direction. When sketch targets have similar cost, the closest to the camera is selected.

Correspondence After the single sketch target is chosen by Eq. (11), exact correspondences between the sketch target S^* and the user sketch C^s need to be identified. The corresponding vertex locations $\{\mathbf{c}_i^s\}$ on this sketch stroke result in constraints on the sketch target $\{\mathbf{s}_i^*\}$ as follows: First, $\{\mathbf{s}_i^*\}$ are transformed to the screen space, i.e., the first two components contain screen space coordinates, while the third component contains the depth value $\{d_i\}$. Then, both curves are parameterized over $[0, 1]$ based on the edge lengths of the target polylines. This induces a mapping from $\{\mathbf{s}_i^*\}$ to $\{\mathbf{c}_i^s\}$, defining new screen space target constraints $\{(\mathbf{c}_i^s, d_i)\}$ that are directly passed to the solver. Note that correspondences are computed in the same way for each sketch target, but the optimization is adapted for each type. In case lengths of the sketch stroke and the sketch target are different, bone lengths are preserved by the IK solver, and different stroke lengths result in spatial warps of the joint path. We will describe details in the following sections.

6.2 Optimization Framework

Given the original motion $\mathcal{M}_0$ and a set of constraints $\mathcal{C}$, we formulate an optimization problem such that a target motion $\mathcal{M} = \mathcal{M}_0 \oplus \boldsymbol{\delta}$ satisfies the sketch constraints $\{(c_i^s, d_i)\}$ in $\mathcal{C}$ by a smooth displacement map $\boldsymbol{\delta}$. To solve this problem in our context, we adopt the hierarchical motion fitting technique of [Lee and Shin 1999]. The final motion $\mathcal{M}$ can be derived by adding a series of successively finer sub-maps $\boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_h}$ which lead to the corresponding series of incrementally refined motions, $\mathcal{M}_1, \ldots, \mathcal{M}_h$ as follows:

$$\mathcal{M}_h = (\cdots(\cdots((\mathcal{M}_0 \oplus \boldsymbol{\delta}_1) \oplus \boldsymbol{\delta}_2)\cdots \oplus \boldsymbol{\delta}_i)\cdots \oplus \boldsymbol{\delta}_h). \qquad (12)$$

Here, $\boldsymbol{\delta}_i \in \mathbb{R}^{3N_j+3}$ $(1 \leqslant i \leqslant h)$, is represented by an array of cubic B-spline curves.

The displacement of the joint at a particular time t is interpolated by the corresponding component curve of the displacement map and thus smoothly propagated to the neighboring frames. At each

constrained time t, we utilize a sketch-based IK solver (Sec. 6.3) with a given set of constraints $C(t) \in \mathcal{C}$ that includes projective constraints imposed by the sketch.

Additionally to the constraints directly imposed by the sketch, we pre-compute all the environmental contact constraints using [Le Callennec and Boulic 2006], and then transform them accordingly to the chosen sketch space. In addition, according to the selection of sketch targets, additional positional constraints (Sec. 4.2) are imposed to prevent inexact movements in unselected body parts. We assume that a specific constraint is defined at a particular instance of time. Constraints on joint paths are considered as a variational constraint [Gleicher 1997] that holds over an interval of motion frames, which is realized by a sequence of constraints for the time interval. For more details on hierarchical motion fitting, please refer to Lee and Shin's work [1999].

Detail control One benefit of the hierarchical displacement mapping technique is that the edited detail can be easily controlled by providing a displacement level h as a user parameter. We map the hierarchy level h in Eq. (12) to the type of the sketching brush that provides either coarse ($h = 1$) or finer control of the result.

6.3 Sketch-based Inverse Kinematics

Our sketch-based IK solver recovers the pose by minimizing an objective function that is a weighted combination of four terms. The first term E_P preserves the original pose, the second term E_S pushes the joint in the selected sketch target towards the corresponding point on the sketch, the third term E_D preserves the original depth of the joint in the selected sketch target, and the fourth term E_C enforces constraints such as joint limitations and environmental contacts,

$$\arg\min_{\mathbf{x}} \omega_P E_P(\mathbf{x}) + \omega_S E_S(\mathbf{x}) + \omega_D E_D(\mathbf{x}) + \omega_C E_C(\mathbf{x}) \quad (13)$$

where the weights ω determine the relative importance of each term, and $\mathbf{x} \in \mathbb{R}^{3N_j+3}$ represents the degrees of freedom of the skeleton. We can parameterize $\mathbf{x}$ from Eq. (4) using the displacement $\boldsymbol{\delta} = (\mathbf{u}, \mathbf{v}_0, \ldots, \mathbf{v}_{N_j-1})$ from a given reference configuration $(\mathbf{p}_0^b, \mathbf{q}_0^b, \ldots, \mathbf{q}_{N_j-1}^b)$ as follows:

$$\mathbf{p}_0 = \mathbf{p}_0^b + \mathbf{u}$$
$$\mathbf{q}_i = \mathbf{q}_i^b \exp(\mathbf{v}_i) \quad (0 \leqslant i < N_j).$$

Penalizing large displacement (E_P) We preserve the original pose by penalizing a large displacement, as commonly practiced in previous research [Gleicher 1997; Gleicher 1998; Lee and Shin 1999],

$$E_P(\mathbf{x}) = \sum_i^{N_j} \alpha_i \|\mathbf{x}_i\|^2, \quad (14)$$

where α_i is used to control the rigidity of an individual joint.

Sketch constraints (E_S) A single sketch stroke automatically produces constraints for a single sketch target. For body lines we use static constraints

$$E_S^{static}(\mathbf{x}) = \sum_{\mathbf{c}_i^s \in C^S} \|\mathbf{c}_i^s - \text{proj}(\mathbf{s}_{j(s)}(\mathbf{x}))\|^2, \quad (15)$$

where C^S is a set of constraints from the sketch, and $\mathbf{s}_{j(s)}(\mathbf{x})$ is a point in the sketch target of joint j that corresponds to a sketch constraint $\mathbf{c}_i^s$.

For joint paths we place constraints at each frame over the editing range of time to describe the continuous relationship between the joint path and the given input. Considering the choice of sketch space —either global, local, or dynamic—, we set a variational constraint on arbitrary time t as follows:

$$E_S^v(\mathbf{x}_t) = \|\mathbf{c}^s(t) - \text{proj}(\mathbf{M}_k(t_c)\mathbf{s}_{j(s)\to k}(\mathbf{x}_t) + \mathbf{w}(t))\|^2, \quad (16)$$

where $\mathbf{c}^s(t)$ is a sketch constraint at time t, $\mathbf{M}_k(t_c)$ is a fixed transformation of joint k at current time t_c (Eq. (6)), $\mathbf{s}_{j(s)\to k}(\mathbf{x_t})$ is a point of joint j in local space with respect to joint k (Eq. (5)), and $\mathbf{w}(t)$ is a warping vector at time t (Eq. (8)).

When $t = t_c$, our local space representation $\mathbf{M}_k(t_c)\mathbf{s}_{j(s)\to k}(\mathbf{x}_t)$ becomes equivalent to $\mathbf{s}_{j(s)}(\mathbf{x}_t)$. That is, $\mathbf{M}_k(t_c)\mathbf{s}_{j(s)\to k}(\mathbf{x}_{t_c}) \equiv \mathbf{s}_{j(s)}(\mathbf{x}_t)$ for any arbitrary joint k. The warping vector is always $\mathbf{0}$ for current time t_c by definition (Eq. (8)). Therefore, we can safely use Eq. (16) for the static case.

In terms of sketch space, frame joint k changes upon switching between global and local spaces. When $k = 0$, $\mathbf{M}_0(t)\mathbf{s}_{j(s)\to 0}(\mathbf{x}_t)$ becomes equivalent to $\mathbf{s}_{j(s)}(\mathbf{x}_t)$ for any arbitrary time t, which represents a point in global sketch space. Therefore, we can use a uniform constraint function both for the static and the variational sketching, and also for every sketch space as follows:

$$E_S(\mathbf{x}_t) = \sum_{c_i^s \in C^S} \|c_i^s(t) - \text{proj}(\mathbf{M}_k(t_c)\mathbf{s}_{j(s)\to k}(\mathbf{x}_t) + \mathbf{w}(t))\|^2. \quad (17)$$

Note that the summation works only for the static case (body line) as a joint path target always has a single target joint for sketching.

Depth constraints (E_D) As it is hard to control depth by sketching, we simply keep the original depth during the sketching process by enforcing a depth constraint as follows:

$$E_D(\mathbf{x}_t) = \sum_{d_i \in C^D} \|d_i(t) - \text{depth}(\mathbf{M}_k(t_c)\mathbf{s}_{j(s)\to k}(\mathbf{x}_t) + \mathbf{w}(t))\|^2, \quad (18)$$

where C^D is a set of depth constraints, and $d(t)$ is a depth constraint at time t. The same principle holds for a depth constraint function in terms of various sketch spaces as in the case of sketch constraints (Eq. (17)).

General constraints (E_C) For the constraints not imposed by the sketch we use

$$E_C(\mathbf{x}_t) = \sum_{\mathbf{c}_i^e \in C^E} \|\mathbf{c}_i^e - (\mathbf{r}_j(\mathbf{x}_t) + \mathbf{w}(t))\|^2 + \sum_{(\theta_i^L, \theta_i^U) \in C^L} \|g(\theta_i^L, \theta_i^U, q_l(\mathbf{x}_t))\|^2, \quad (19)$$

where C^E and C^L are a set of constraints computed from the environment contacts and joint limits. $\mathbf{r}_j(\mathbf{x})$ and $q_l(\mathbf{x})$ are the position and the angle of joint j and l, respectively. $g(\cdot)$ is a modified version of the smooth max function that penalizes the range outside (θ^L, θ^U) [Tassa et al. 2012].

IK optimization To obtain IK solutions for every constrained time frames at interactive rates, we analytically evaluate the Jacobian terms of the objective function in Eq. (13). As our optimization problem is nonlinear due to the projective division in Eq. (17), we opt for the quasi-Newton L-BFGS [Nocedal and Wright 2006; Johnson 2010] method.

7 Extensions

In this section, we demonstrate how the functionality of SketchiMo can be easily extended to address various editing scenarios by building directly on top of the core functionality detailed in the previous

sections. We first extend sketch targets to include an abstract visualization of the motion, and then explain how the sketch space and optimization can be modified to achieve the different motion solutions for relative re-timing and noise removal. These modifications are incorporated into the SketchiMo framework and are presented to the user as additional sketch targets and brushes.

7.1 Abstract Sketch Targets

We extend the definition of the body line and the joint path to obtain a meaningful abstraction of the given motion. Motion abstraction has proven useful for high-level manipulation or the extraction of motion features [Assa et al. 2005]. The motion path [Gleicher 2001] is a well known abstraction of the motion that can be obtained by projecting the translational motion of a character onto the ground. We target our abstraction towards the manipulation of coordinated motion.

Generally, we consider an abstract target as any polyline derived from motion properties that does not directly lie on the body and that is not coincident with a joint trajectory. We explicitly define two such abstract targets, the *average path* and the *relational line*. Note that the motion path could also be easily incorporated into the collection of abstract targets available in our framework.

An **Average Path** represents the common trajectory of selected joints. By editing the average path, several joint trajectories can be edited at once while preserving the properties of the original separate joint paths. The average path is defined as follows:

$$S^{ap} = \left\{ \mathbf{s}^{ap}(t) \in \mathbb{R}^3 \;\middle|\; \mathbf{s}^{ap}(t) = \tfrac{1}{N(J^{sel})} \sum_{j \in J^{sel}} \mathbf{s}_j^{jp}(t) \right\}, \quad (20)$$

where J^{sel} represents a set of joints currently selected, $N(J^{sel})$ is the number of the selected joints, and $\mathbf{s}^{ap}(t)$ is the average point at time t, respectively.

A constraint function for the average path includes an offset vector $\mathbf{o}_j(t) = \mathbf{s}^{ap}(t) - \mathbf{s}_j^{jp}(t)$ as follows:

$$E^{ap}(\mathbf{x}_t) = \sum_{j \in J^{sel}} \Big(\|\mathbf{c}^s(t) - \text{proj}(\mathbf{s}_j(\mathbf{x}_t) + \mathbf{w}(t) + \mathbf{o}_j(t))\|^2 + \|d(t) - \text{depth}(\mathbf{s}_j(\mathbf{x}_t) + \mathbf{w}(t) + \mathbf{o}_j(t))\|^2 \Big). \quad (21)$$

The solution to the average path can be obtained from the minimization of Eq. (13) by replacing E_S and E_D with Eq. (21).

A **Relational Line** visualizes the relationship between selected end-effectors by connecting them through space with a linear polyline:

$$S^{rl} = \left\{ \mathbf{s}_j^{rl}(t) \in \mathbb{R}^3 \;\middle|\; \forall j \in J^{sel} \right\}. \quad (22)$$

Unlike a body line (Eq. (1)), the selected joints J^{sel} in a relational line do not necessarily lie on a joint chain J^{chain}. Eq. (17) and (18) are directly used to constrain the optimization of the relational line.

7.2 Re-timing

A significant amount of time is spent adjusting the timing of an animation. We aim to exploit the visual feedback on speed given by the joint paths to devise a re-timing tool that operates by sketching directly on the joint path. By brushing over a certain section of the path, the user can speed up, slow down, or linearize time (i.e. evenly distribute frames, see Fig. 9).

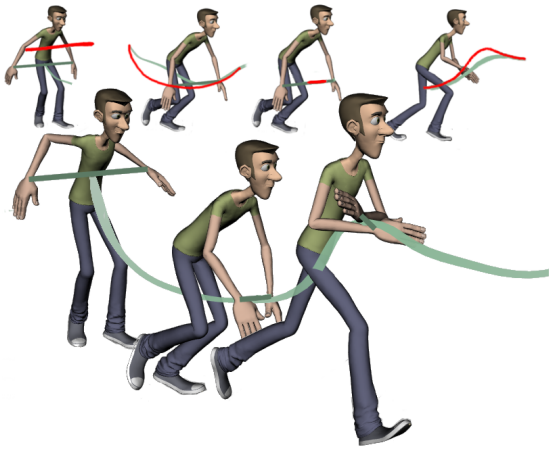

Figure 8: *Abstract targets. The average path (light green) sweeps from the center of the relational line (dark green) linking the selected joints. The edited motion (bottom) was obtained from the highlighted edits (red) shown over the original motion (top).*

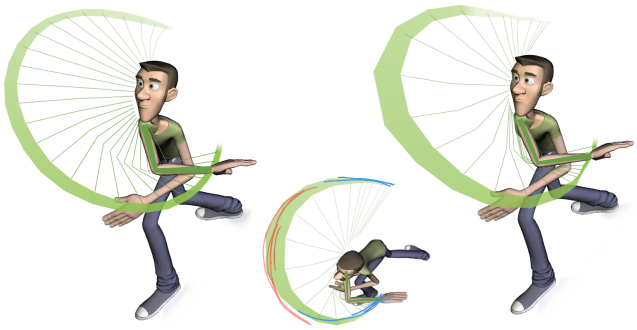

Figure 9: *Re-timing. Left: original motion. Center: user strokes to speed up (red) and slow down (blue) the motion. Right: edited path.*

A change of the speed in the entire path can be obtained by re-parameterizing the joint path as follows:

$$\arg\min_{P} \sum_{p_i \in P} \|\sigma_i - \Delta(p_i)\|^2 + \sum_{p_s \in P^S} \|w\sigma_s - \Delta(p_s)\|^2$$
$$\text{s.t.} \quad p_0 = 0.0,\ p_{N_p - 1} = 1.0,\ P^S \subset P. \quad (23)$$

P^S and P are a set of ordered parameter values of the sketched and the bounded region of the selected joint path. σ is the original distance between two parameters, measured along the path, and w is the weight value given by the user sketch inputs, respectively. A value $w < 1.0$ will slow down the brushed motion, while setting $w > 1.0$ will increase the speed in the area. $\Delta(p_i) = p_i - p_{i-1}$ is simply the finite difference between two parameters.

The new positions $\mathbf{s}(p_i^*) \in \mathbb{R}^3$ obtained from the $1D$ Laplacian in the path parameter space are used as soft constraints for the optimization. As the positions of the constraints are computed by re-parameterizing the original path, the shape of the path does not change, only the timing is altered. It is sometimes useful to linearize the motion by evenly redistributing frames over the trajectory. This is easily implemented by directly specifying the uniform distribution over the sketched region as constraints for the optimization.

An important factor to consider is the careful selection of the temporal boundaries for the optimization. The re-timing optimization

will produce a compensation effect where speeding up a region will slow down the remaining parts. If correct boundaries are not carefully chosen, the re-timing can introduce visible timing artifacts. We extend the contextual boundary selection of the sketch space by replacing the view-dependent boundary selection in Eq. (9) with the consideration of the speed of the selected sketch target:

$$T_{sp}^{-} = \{t_c - m, \ldots, t_c \mid |\dot{\mathbf{s}}(t_c - m)| < \nu\},$$
$$T_{sp}^{+} = \{t_c, \ldots, t_c + n \mid |\dot{\mathbf{s}}(t_c + n)| < \nu\}, \quad (24)$$

where $|\dot{\mathbf{s}}(t)|$ is the speed of the selected joint path at time t, and ν is a user parameter bounding the maximum stopping speed of the path. Starting from the current frame, the sketch space will search for velocity inflection points on the trajectory to delimit the editing range. This has the additional benefit of fixing the sketch space within a certain time range. Although we present a visual speed cue through the joint path thickness, checking the animation is necessary. The user can play the animation back and forth while having visual cues for the speed and spatial trajectory.

7.3 Noise Removal

A similar technique can be utilized to remove noise from the motion data. Here, instead of a $1D$ Laplacian on the parameter space, we apply the Laplacian to the $3D$ path positions $L(\mathbf{s}(t)) = \mathbf{s}(t) - \frac{1}{2}(\mathbf{s}(t-1) + \mathbf{s}(t+1))$ to smooth out the path:

$$\arg\min_{S} \sum_{\mathbf{s}(t)\in S} \|\boldsymbol{\sigma}(t) - L(\mathbf{s}(t))\|^2 + \sum_{\mathbf{s}_r \in S^R} \|\mathbf{s}'_r - \mathbf{s}_r\|^2 + \sum_{\mathbf{s}_s \in S^S} \|wL(\mathbf{s}_s)\|^2, \quad s.t. \quad S = S^S \cup S^R, \quad (25)$$

where S^S and S^R denote the set of ordered path positions closest to the given sketch and the remaining region of the selected joint path S, respectively. $\boldsymbol{\sigma}$ is the original Laplacian coordinate of the points in the selected sketch target. The original joint path positions $\mathbf{s}'_r$ are used to constrain points outside the sketched region. Control over smoothness can be achieved with the user defined scaling factor w. SketchiMo visualizes the smoothing power by proportionally varying the smoothing brush size. The edited motion is solved by constraining the optimization with the smoothed positions $\mathbf{s}_i \in \mathbb{R}^3$.

8 Results

8.1 Experimental Setup

Implementation details We prototyped SketchiMo as a standalone application using Qt 5.5 for the graphical user interface, and GLSL for the visualization. For the sketching inputs, we offered three different brushes, the *line* brush to specify new positions for the sketch targets, the *re-timing* brush to speed up, slow down, or linearize the velocities, and the *smoothing* brush to remove unwanted details. For the line and smoothing brushes, the brush size regulates the editing detail (Eq. (12)) and smoothness power (Eq. (25)), respectively. The re-timing brush is visualized with the help of an additional reference circle. A relatively bigger, matching, or smaller size will switch between speeding up, evenly spreading, or slowing down behaviors. We also exposed the choice of sketch space, and its tuning parameters, i.e., the projection (Eq. (10)) and the speed threshold (Eq. (24)), the dynamic warping factor (Eq. (7)), and the maximum number of frames to be bounded within the sketch space. For the IK optimization, we fixed $\omega_P = 0.01, \omega_S = 100, \omega_D = 1, \omega_C = 1$ from Eq. (13).

Example data We experimented with three different characters. The *Malcom* (Fig. 1) and *Gump* (Fig. 11) models share the same skeletal structure consisting of 65 joints and 115 DoFs, but have different proportions. The *Pink Panther* (Fig. 13) has a different structure including a tail and consists of 28 joints and 57 DoFs. We only allow joint rotations of the skeleton with the exception of the root that has three translational DoFs. To reduce a total number of DoFs, the neck and the clavicles are constrained to two DoFs, and the knee, the elbow, and the finger joints are constrained to a single DoF. We manually specified joint angle limits. The motion data used in our experiments were captured from a Vicon optical system, or obtained from pre-existing motion databases [Autodesk a]. All of the motion data clips were retargeted to the characters with a sample rate of 60 fps.

8.2 Experimental Results and Discussion

We present six edited motion sequences created by five users (including two authors) with different skill levels ranging from novice to five years experience in character rigging. All the results presented were made entirely via sketch inputs. We started the experiments by presenting the user with a motion clip and an editing goal for the sequence, then, we let the user sketch all the necessary strokes until the goal was accomplished.

Use-case examples The combination of fast and flexible sketch target selections and the choice of sketch space support editing of various types of movements. The role of the sketch space is illustrated in Fig. 4 and 5. Fig. 4 shows how a complex global motion becomes simple arcs in the local space. With a single stroke, the forward and backward swing of the left arm is made wider. By drawing the ending of the swing closer to body, the hand is brought

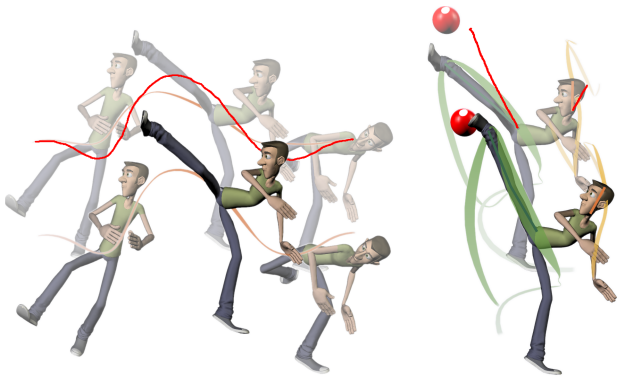

Figure 10: *"Taekwondo" motion editing to hit a given target (red ball). The root motion is first edited in the dynamic space (left). Body line edits are then added on the kick pose (right).*

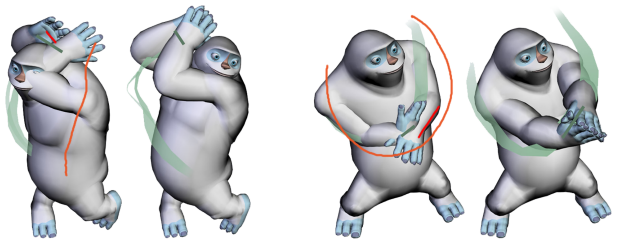

Figure 11: *Correcting retargeting artifacts on the "Golf" motion. Left: Pose edits to fix interpenetrations. To avoid visual clutter only the user input on the spine body line is shown (orange). The neck was also edited in another view. Right: The hand contact is fixed (red), the trajectory is then modified into a clean arc (orange).*

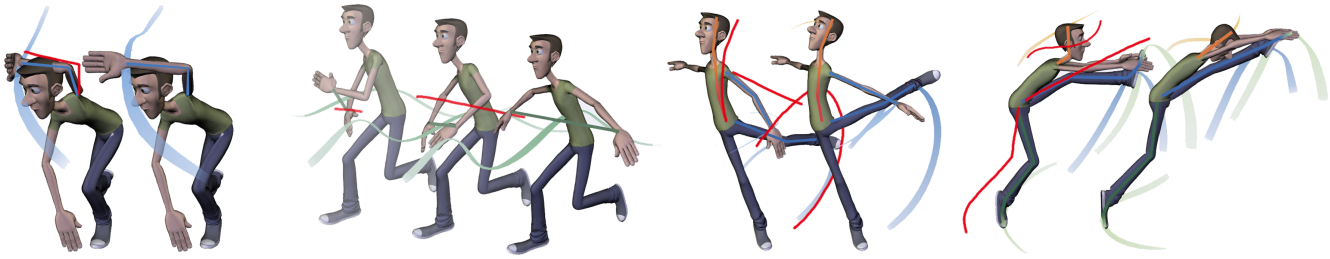

Figure 12: *Example edits on the "Ballet" motion. Left: Correction of the interpenetration. Center left: Users were asked to uncross the hands during a portion of the motion. Center right: The expressiveness of the motion is enhanced by specifying a body line and modifying the ankle joint path. Right: In addition to exaggerating the pose, a nodding motion is added to the head.*

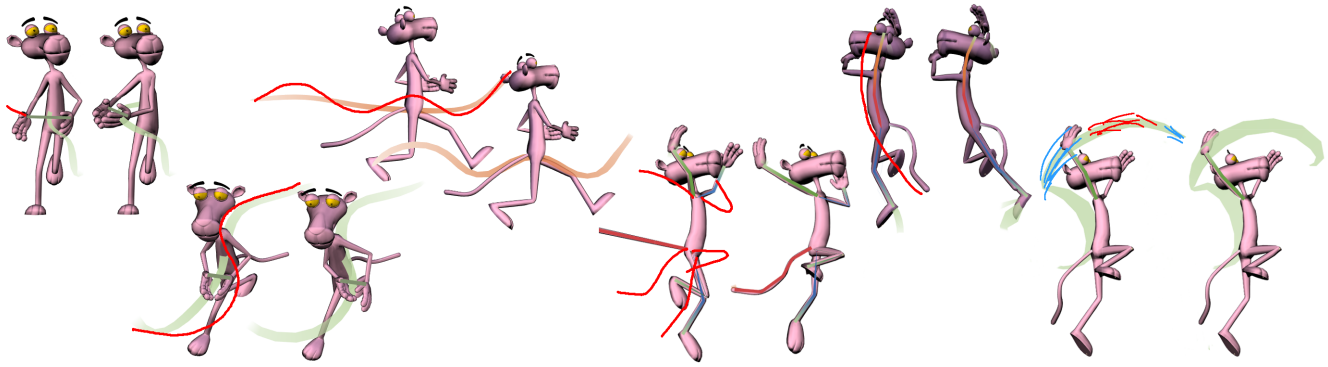

Figure 13: *Sketch inputs during the editing of the "Dunk" motion. Top, from left to right: A pose edit using a relational line, exaggerating motion by editing the root joint path, enhancing the expressiveness of the dunk pose by sketching on a head to foot body line. Bottom, from left to right: Coordinated hand motion is jointly edited with an average path, multiple limb body lines to specify a pose, re-timing edits to exaggerate the dunking motion. (Pink Panther character courtesy of Karim Kashefi)*

closer to the chest. Ensuring smooth arcing motions and exaggerating the anticipation and follow-through of the motions is one of the main benefits of this space. Fig. 5 presents how the dynamic space stretches the trajectory along the horizontal axis, thereby disambiguating the spatially overlapping motion and allowing accurate visualization and editing of vertical movement. In Fig. 5 bottom, all the crouching motions are exaggerated with a single stroke in the dynamic space. Fig. 8 and 11 demonstrate the flexibility of the abstract targets for the editing of coordinated motion. In Fig. 8, with four simple sketches, a crossing hands motion is enhanced by increasing the anticipation and follow-through, as well as ensuring a clean arcing trajectory. Fig. 9 illustrates how a few strokes are enough to enhance the easing in and out of the arm swing using the re-timing brush.

Practical motion editing scenarios In Fig. 10 ("Taekwondo"), the user was given a target contact position (red ball) to kick. The target is placed out of the reach of the original motion, requiring not just the editing of the kicking pose but also the jump. As the jump is mostly done in place, the root joint path was edited in the dynamic space in order to exaggerate the anticipation, height and landing of the jump, portraying a stronger effort. Finally, the body lines of the leg and head were edited in order to hit the target and correct the gaze direction, respectively. Fig. 11 ("Golf") demonstrates a common editing situation arising when retargeting motion to characters with significantly different proportions. We asked users to fix the motion artifacts resulting from the retargeting —mostly interpenetrations and the lack of hand contacts (the character is supposed to be grabbing a golf club). With a few edits the motion is correctly adapted to the new character. Fig. 13 ("Dunk") and 12 ("Ballet") represent typical motion editing scenarios. These particular examples involve many types of movements, including running, jumping and dribbling a basketball in a single clip. The users were asked to fix motion interpenetrations, achieve certain given poses, and overall to increase expressiveness of the motion. Fig. 13 illustrates a few edits on the "Dunk" motion. To edit the coordinated motion of the hands grabbing the ball, the relational line was used to first exaggerate two extreme poses by displacing the hands towards the side. Then the average path was edited to avoid the interpenetration generated after the previous mentioned edit. It was also used to exaggerate the swinging to the side. The running motion was exaggerated with a simple stroke on the root path. A few body lines were sufficient to increase the drama of the poses.

Motion	Frames	Spaces	Num. of Strokes				Time
			bl	jp	at	r/s	
Martial Arts	283	g/d	2	3	2	3	2 m. 30 s.
Parkour	600	g/d	2	1	0	0	3 m. 20 s.
Taekwondo	690	g/d	5	3	0	2	4 m. 30 s.
Golf	636	g	6	2	10	2	7 m. 40 s.
Dunk	364	$g/l/d$	12	5	3	2	9 m. 50 s.
Ballet	1320	$g/l/d$	10	8	3	0	6 m. 30 s.

Table 1: *Interaction breakdown of a single novice user: sketch spaces (global, local, dynamic), total strokes (body line, joint path, abstract target, re-timing/smoothing), and editing time.*

Table 1 presents a breakdown of the user interactions over the different editing tasks. We instrumented Sketchimo to record the total number of given strokes, as well as how they were distributed within the different types of sketch targets and spaces. The total number of time spent in SketchiMo in order to achieve the desired editing is also presented, demonstrating that users with varied skill

levels can accomplish effective motion editing in a very reasonable timeframe. The users often preferred to switch between selection rather the keeping multiple selected targets on the viewport. In general, the editing process was approached by prioritizing pose edits through body lines in the global space and switching to joint paths at a later stage in order to control the in-betweening of the modified poses. The local space was left mostly untouched until the later stages of the editing process when more fine control was necessary.

Computation time SketchiMo supports fast, interactive editing. We ran our experiments on a desktop computer, having an Intel i7-3770 @ 3.40 GHz and 8 GB RAM with a NVidia GT640 graphics card. The visualization of the characters and the sketch space runs at 120 fps with 20 selected sketch targets on the screen. For 600 frames (10 sec.) of motion, the optimization required 1200 ms to compute. For 60 frames (1 sec.), the motion is solved in 100 ms.

9 Limitations and Future Work

To ease the editing and to preserve important motion features, we handle constraints automatically. Nevertheless, in order to introduce significant changes, direct control over constraints is necessary. Letting the user specify additional static constraints would be useful when editing interactions with objects or the environment. More advanced constraints could also be considered in order to constrain different aspects of the motion such as the shape of the body line [Guay et al. 2013] or to preserve physical properties [Popović and Witkin 1999] such as velocity and acceleration of the motion.

Body lines are a versatile solution for specifying the pose of the character, but control over orientation is limited. While editing the bending orientation of 1- and 2-DoFs body parts can be effectively addressed with body lines, manipulation of axial rotation of 3-DoFs joints is not possible in the current implementation.

SketchiMo does not support extensive manipulations of timing. Consider the case of extending a walking motion for a few meters. Applying a dynamic time warping would ease the problem to a certain extend, but would inevitably degenerate into unnaturally long strides. It would be necessary to consider a discrete optimization procedure to introduce the additional motion samples [Kim et al. 2009]. In addition, although we work with densely sampled motion data, extensive editing can eventually result in poor sample spacing. We plan to incorporate a resampling strategy, e.g. by fitting spline curves to the result after each edit.

Our system successfully handles abstract concepts such as the motion path, coordinated motion, and relative relationships. Additional extensions could be envisaged to address the synchronization and editing of multiple characters. We also plan to explore brushing behaviors related to signal processing of the data such as exaggerating motion properties for semi-automated stylization.

In the future we plan to address the creation of animation by specifying the motion directly with sketches, rather than editing it. Guay et al. [2015b] demonstrated that it is possible to animate relatively simple characters by specifying dynamic line of actions with a single gesture. Extending their achievement to articulated characters with multiple environmental contacts remains an ambitious problem with significant challenges.

10 Conclusion

Successful sketch-based editing relies on the choice of a small but powerful set of interface constructs. Previous research has demonstrated the value of constructs such as motion paths [Gleicher 2001], the line of action [Guay et al. 2013], and the spacetime curve [Guay et al. 2015b].

In this paper we presented SketchiMo, a framework that can accomplish entirely sketch-based motion editing of full body characters. SketchiMo is founded on the twin constructs of sketch targets and sketch spaces. Together, these constructs are sufficient to naturally disambiguate the role of each stroke. They are interpreted through an underlying optimization formulation that naturally incorporates sketched constraints from different sketch spaces and targets in a unified formulation. Unlike some previous research, there is no requirement that strokes be entered in any particular order.

The result is simple, expressive editing of all aspects of an animation, including body poses, global and local trajectories, spatial relationships, and timing. We demonstrated that SketchiMo is a versatile platform for expressive motion editing in production scenarios, including the correction of retargeting artifacts from motion capture, stylization of animation, and in-betweening of keyframe animation.

Acknowledgements

We thank the anonymous reviewers for their valuable comments; Jaewon Song and Junghee Kim for the helpful discussion about the animation process; Daseong Han for his valuable feedback on the physics-based animation; Hyungjin Kim and Minjeong Shin for modeling the Gump character. This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2014R1A2A1A01002871).

References

ANNETT, M., ANDERSON, F., BISCHOF, W. F., AND GUPTA, A. 2014. The pen is mightier: Understanding stylus behaviour while inking on tablets. In *Proceedings of GI '14*, 193–200.

ASSA, J., CASPI, Y., AND COHEN-OR, D. 2005. Action synopsis: Pose selection and illustration. *ACM Trans. Graph. 24*, 3 (July), 667–676.

AUTODESK. Maya, MotionBuilder products. www.autodesk.com.

AUTODESK. Mudbox. www.autodesk.com/mudbox.

BLENDER FOUNDATION. Blender. www.blender.org.

BOROSÁN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. 2012. Rigmesh: Automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph. 31*, 6 (Nov.), 198:1–198:9.

BRAND, M., AND HERTZMANN, A. 2000. Style machines. In *Proceedings of SIGGRAPH '00*, 183–192.

CHOI, M. G., YANG, K., IGARASHI, T., MITANI, J., AND LEE, J. 2012. Retrieval and visualization of human motion data via stick figures. *Comput. Graph. Forum 31*, 7pt1 (Sept.), 2057–2065.

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. In *Proceedings of SCA '08*, 137–146.

DE PAOLI, C., AND SINGH, K. 2015. Secondskin: Sketch-based construction of layered 3d models. *ACM Trans. Graph. 34*, 4 (July), 126:1–126:10.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings of I3D '97*, 139–148.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of SIGGRAPH '98*, 33–42.

GLEICHER, M. 2001. Motion path editing. In *Proceedings of I3D '01*, 195–202.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Trans. Graph. 23*, 3 (Aug.), 522–531.

GUAY, M., CANI, M.-P., AND RONFARD, R. 2013. The line of action: An intuitive interface for expressive character posing. *ACM Trans. Graph. 32*, 6 (Nov.), 205:1–205:8.

GUAY, M., RONFARD, R., GLEICHER, M., AND CANI, M.-P. 2015. Adding dynamics to sketch-based character animations. In *Proceedings of SBIM '15*, 27–34.

GUAY, M., RONFARD, R., GLEICHER, M., AND CANI, M.-P. 2015. Space-time sketching of character animation. *ACM Trans. Graph. 34*, 4 (July), 118:1–118:10.

HAHN, F., MUTZEL, F., COROS, S., THOMASZEWSKI, B., NITTI, M., GROSS, M., AND SUMNER, R. W. 2015. Sketch abstractions for character posing. In *Proceedings of SCA '15*, 185–191.

HO, E. S. L., KOMURA, T., AND TAI, C.-L. 2010. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph. 29*, 4 (July), 33:1–33:8.

HSU, E., PULLI, K., AND POPOVIĆ, J. 2005. Style translation for human motion. *ACM Trans. Graph. 24*, 3 (July), 1082–1089.

HSU, E., DA SILVA, M., AND POPOVIĆ, J. 2007. Guided time warping for motion editing. In *Proceedings of SCA '07*, 45–52.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH '99*, 409–416.

JOHNSON, S. G., 2010. The nlopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. *ACM Trans. Graph. 24*, 3 (July), 934–934.

KIM, M., HYUN, K., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. *ACM Trans. Graph. 28*, 3 (July), 79:1–79:9.

LASSETER, J. 1987. Principles of traditional animation applied to 3d computer animation. *Comput. Graph. 21*, 4 (Aug.), 35–44.

LE CALLENNEC, B., AND BOULIC, R. 2006. Robust kinematic constraint detection for motion data. In *Proceedings of SCA '06*, 281–290.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH '99*, 39–48.

LEE, J. 2008. Representing rotations and orientations in geometric computing. *IEEE Comput. Graph. Appl. 28*, 2 (Mar.), 75–83.

LEVI, Z., AND GOTSMAN, C. 2013. ArtiSketch: A system for articulated sketch modeling. *Comput. Graph. Forum 32*, 2pt2 (May), 235–244.

LIN, J., IGARASHI, T., MITANI, J., AND SAUL, G. 2010. A sketching interface for sitting-pose design. In *Proceedings of SBIM '10*, 111–118.

MCLAUGHLIN, T., CUTLER, L., AND COLEMAN, D. 2011. Character rigging, deformations, and simulations in film and game production. In *SIGGRAPH '11 Courses*, 5:1–5:18.

MUKAI, T., AND KURIYAMA, S. 2009. Pose-timeline for propagating motion edits. In *Proceedings of SCA '09*, 113–122.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3 (July), 1142–1147.

NEFF, M., AND FIUME, E. 2003. Aesthetic edits for character animation. In *Proceedings of SCA '03*, 239–244.

NOCEDAL, J., AND WRIGHT, S. 2006. *Numerical optimization*. Springer Science & Business Media.

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 SCA '13*, 155–164.

PIXOLOGIC. Zbrush. www.pixologic.com.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH '99*, 11–20.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 4 (Oct.), 1034–1054.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5 (Sept.), 32–40.

SCHMID, J., SENN, M. S., GROSS, M., AND SUMNER, R. W. 2011. Overcoat: An implicit canvas for 3d painting. *ACM Trans. Graph. 30*, 4 (July), 28:1–28:10.

SHAPIRO, A., CAO, Y., AND FALOUTSOS, P. 2006. Style components. In *Proceedings of GI '06*, 33–39.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4 (July), 97:1–97:8.

TASSA, Y., EREZ, T., AND TODOROV, E. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. *Intelligent Robots and Systems* (Oct.), 4906–4913.

TERRA, S. C. L., AND METOYER, R. A. 2004. Performance timing for keyframe animation. In *Proceedings of SCA '04*, 253–258.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Trans. Graph. 23*, 3 (Aug.), 424–431.

WANG, J., DRUCKER, S. M., AGRAWALA, M., AND COHEN, M. F. 2006. The cartoon animation filter. *ACM Trans. Graph. 25*, 3 (July), 1169–1173.

WEI, X., AND CHAI, J. 2011. Intuitive interactive human-character posing with millions of example poses. *IEEE Comput. Graph. Appl. 31*, 4 (July), 78–88.

WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH '95*, 105–108.

YOO, I., VANEK, J., NIZOVTSEVA, M., ADAMO-VILLANI, N., AND BENES, B. 2014. Sketching human character animations by composing sequences from large motion database. *Vis. Comput. 30*, 2 (Feb.), 213–227.

YOO, I., MASSIH, M. A., ZIAMTSOV, I., HASSAN, R., AND BENES, B. 2015. Motion retiming by using bilateral time control surfaces. *Computers & Graphics 47*, 59–67.