Authoring Motion Cycles

Loïc Ciccone
ETH Zurich

Martin Guay
Disney Research Zurich

Maurizio Nitti
Disney Research Zurich

Robert W. Sumner
ETH Zurich, Disney Research Zurich

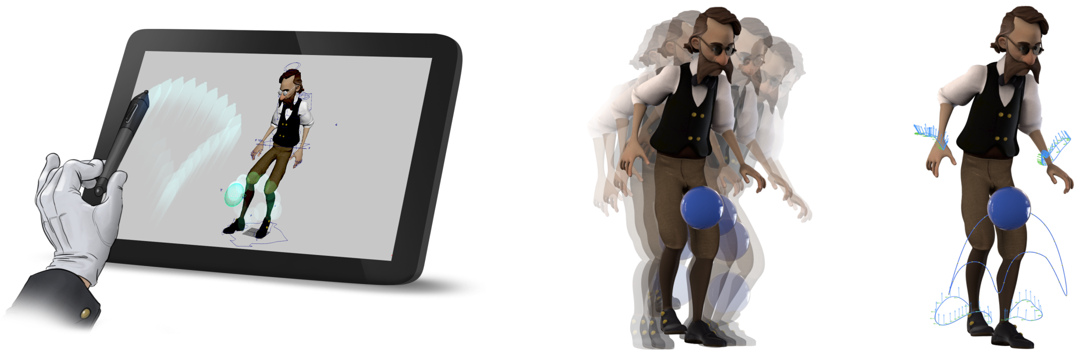

Figure 1: Left: To specify a motion cycle, the user acts out several loops of the motion using a variety of capture devices. Middle: A looping motion cycle is automatically extracted from the noisy performance. Right: A custom motion representation tool, called MoCurves, allows controlling and coordinating spatial and temporal transformations from a single viewport.

ABSTRACT

Motion cycles play an important role in animation production and game development. However, creating motion cycles relies on general-purpose animation packages with complex interfaces that require expert training. Our work explores the specific challenges of motion cycle authoring and provides a system simple enough for novice animators while maintaining the flexibility of control demanded by experts. Due to their cyclic nature, we show that performance animation provides a natural interface for motion cycle specification. Our system allows the user to act several loops of motion using a variety of capture devices and automatically extracts a looping cycle from this potentially noisy input. Motion cycles for different character components can be authored in a layered fashion, or our method supports cycle extraction from higher-dimensional data for capture devices that deliver many degrees of freedom. After capture, a custom curve representation and manipulation tool allows the user to coordinate and control spatial and temporal transformations from a single viewport. Ground and other planar contacts are specified with a single sketched line that adjusts a curve's position and timing to establish non-slipping contact. We evaluate the effectiveness of our work through tests with both novice and expert users and show a variety of animated motion cycles created with our system.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; *Graphics systems and interfaces*;

KEYWORDS

Motion cycles, Animation interface, Performance, Motion curves

ACM Reference format:
Loïc Ciccone, Martin Guay, Maurizio Nitti, and Robert W. Sumner. 2017. Authoring Motion Cycles. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 9 pages.
DOI: 10.1145/3099564.3099570

1 INTRODUCTION

Motion cycles play an important role in animation and games. In animated films and visual effects, artists use walk cycles and other looping animations during character development to explore a character's particular movement style. During production, these cycles aide the animation workflow by providing a starting point for walking, running, and other cyclic movements. In games, motion cycles play an even more prominent role. Motion cycles for locomotion allow characters to move arbitrarily under the player's direction. Punching, kicking, flipping, and countless other cycles are created to enable game combat. Even idle game characters are animated with motion cycles for breathing and other subtle movements that give them life. In fact, motion cycles play such an important role that game engines have specialized animation components designed to play back and seamlessly blend between different motion cycles.

Although specialized components for *using* motion cycles are commonplace, little work has explored the specific challenges of *authoring* them. Instead, the authoring process relies on general-purpose animation packages such as Autodesk Maya, 3ds Max, or Blender. These packages, by design, accommodate a broad spectrum of animation tasks with support for features and workflows used in animated films, visual effects, and games. This generality comes at a tremendous cost. The learning curve for any commercial animation package is steep, and the animation process relies on complex mechanisms that require expert knowledge to master. As a consequence, although a motion cycle may only be a few frames of repeated animation, creating those frames with general-purpose animation software is difficult, time consuming, and restricted to expert users.

Our work explores the challenge of motion cycle authoring and provides a system simple enough for novice animators while maintaining the flexibility of control demanded by experts. Our method relies on the core observation that the cyclic nature of motion cycles makes them especially appropriate for performance animation. Our system allows the user to act several loops of motion using a variety of capture devices ranging from the computer mouse to a full body motion-capture suit. Since these acted cycles will inevitably contain imprecisions, we propose an optimization algorithm to analyze and automatically extract a looping cycle from this potentially noisy input. By supporting multidimensional input, cycles can accommodate an arbitrary number of animation variables. We then introduce *MoCurves*, a representation and manipulation tool for motion cycles that encompasses translation, rotation, scale and time. MoCurves allow the user to control and coordinate spatial and temporal transformations from a single viewport. Motion cycles for different character components can be authored independently in a layered fashion and synchronized in time using an optimization-based time-warping function built into the *MoCurve* interface. Finally, since contact with the ground or other surfaces is pervasive in motion cycles yet difficult to precisely control with a performance-based interface, we support a sketch based contact specification in which a single sketched contact line induces a spatio-temporal transformation that respects planar contacts without sliding.

Our work introduces an effective system tailored to motion cycles authoring. Our core technical contributions include a generic motion cycle extraction algorithm, the *MoCurve* representation with support for coordinated spatial and temporal editing, and a contact specification method that uses a single sketched line to establish non-slipping planar contacts. We implemented our approach as an Autodesk Maya plugin that permits motion cycle authoring independent of Maya's more complex animation features. We evaluated the effectiveness of our work through tests with both novice and expert users and showed motion cycles created with performance input from the mouse, Wacom Cintiq tablet, Leap Motion hand tracker, HTC Vive, as well as full-body motion capture. All these elements show that the creation process can be dramatically simplified using our software, allowing novice animators to author quality animations in minutes.

2 RELATED WORK

Keyframing. consists of specifying poses at specific moments in time, and manipulating temporal curves that interpolate different rig parameters. Despite being the most common approach for authoring motions, keyframing is challenging as it requires continuously switching between three different interfaces: the viewport for character posing, the timeline for timing and the graph editor for manipulating curves. In contrast, our approach merges the different controls into a single curve interface (we call MoCurves) that allows one to synergistically control both space and time while remaining in the comfort of a single viewport.

To Pose or to Time? Many works specifically aim to ease the posing process using 2D stick figures [Choi et al. 2012; Lin et al. 2010; Wei and Chai 2011], lines of action [Guay et al. 2013; Öztireli et al. 2013] or other sketch-based abstractions [Ciccone et al. 2016; Hahn et al. 2015] and others focused specifically on improving the timing process through time warping [Coleman et al. 2008; Hsu et al. 2007; Witkin and Popovic 1995]. These methods, however, are only focused on either spatial or temporal editing, and do not help the coordination process. One way to improve coordination is through visualization of the motion. Recently, the de facto timeline visualization has been questioned and enhanced with editable pose-icons representing the motion over the timeline [Mukai and Kuriyama 2009] and deformable spatial planes rendered directly onto the viewport [Yoo et al. 2015]. In our work, the MoCurves interface allows the visualization of several aspects of the motion as well as their coordinate manipulation.

Performance animation. As humans have an instinctive sense of movement and timing, performance animation has been used as a natural way of specifying motions, either for the full character [Chai and Hodgins 2005; Igarashi et al. 2005; Kim et al. 2013; Tautges et al. 2011], or through a layered performance type of approach [Choi et al. 2008; Dontcheva et al. 2003; Jin et al. 2015; Martin and Neff 2012; Neff et al. 2007], with a few works focused on retiming existing motions using gestures [Terra and Metoyer 2007; Walther-Franks et al. 2012]. One problem with performance animation is that when gesturing cyclic motions, users are not accurate enough to perfectly close the loop. Instead, they must rely on manual tools to extract a clean loop. In our work, we make the important observation that performance animation for motion cycles is most natural when the motion is repeated several times. We then propose an algorithm to deliver a single closed loop from a sequence of loops performed by the user.

Cyclification. One core contribution of our work is the automatic cyclification of an acted motion. While some existing research [Ahmed et al. 2003; Mukai 2011; Rose et al. 1996] explores cyclification, these methods operate under the single-cycle assumption and cannot accommodate cyclification of repeated motions. They focus on matching boundaries of a single cycle and do not allow identifying a period in a longer sequence of an imprecisely repeated motion. In Rose et al. [1996], the user is even asked to manually specify the start and end points of a cycle. In our case, taking multiple loops as input is a technical challenge as it requires identifying a recurrent pattern over imprecise loops in space and time. The method in [Silva et al. 1999] can take as input multiple loops, but can only process 1-dimensional curves. In contrast, we solve for N dimensions, which allows us to find the best period for the N-dimensional signal directly. Moreover, similar to their approach, we experimented with frequency domain decomposition and found that, for performance animation, the input is considerably noisy in the temporal dimension (i.e. all acted cycles do not have the same duration), which prevents us from using frequency analysis such as Fast Fourier Transform, and led to our feature-based solution.

Gesturing space-time curves. Close to our work is the concept of gesturing space-time curves for authoring [Guay et al. 2015] or editing existing motions [Choi et al. 2016]. Guay et al. [2015] are able to create a full character motion using a single stroke and refine it using additional types of stroke edits. Unfortunately, their strokes are designed around specific types of motions such as bouncing, rolling and waving, and their work only demonstrates a few simple characters (e.g. no bipeds nor quadrupeds). Choi et al. [2016] allow editing a wide range of motions using sketches, but their work is restricted to editing and cannot author new motions. Using screen-space strokes to define 3D deformations, these two papers end up with an underconstrained problem and thus need to make assumptions about the user's intentions. Our work strives to give the artist freedom over the animation process with minimal restrictions. We moreover retain a high degree of genericity so that motion cycles can be applied to any animation control, such as IK handles, bone transformations, or even vertices.

Animation tools for casual users. Many works seek to generate animations through simple interfaces. Motion data can be used to solve for underconstrained interfaces, such as abstract motion doodles [Thorne et al. 2004] or point trajectories [Jeon et al. 2010; Min et al. 2009; Yoo et al. 2014]. Another idea is to use simulated mechanics and to parameterize controllers w.r.t. to a direction [Coros et al. 2010; Hodgins et al. 1995; Laszlo et al. 2000; Yin et al. 2007]. By construction, these methods restrict the scope of possible movements with the range of preexisting motions (often humanoid motion-capture) or to specific motions such as biped and quadruped locomotion. In contrast, our approach provides fine-scale control of the animation, making it possible to create arbitrary movements for any type of character.

3 OVERVIEW AND WORKFLOW

Our system is designed to create, represent and manipulate cyclic animations in a natural way. To create a motion cycle, the user performs the motion using any capture device, such as a mouse, Leap Motion, HTC Vive or full body motion capture suit (as shown in our Results Section 6). Depending on the device being used, the user may choose to perform the whole character motion or to animate parts of the character in a layered fashion. Also, since synchronizing several motions is a crucial element of animation, we play the animation of all previously created motions while the user performs for other items (or other transformations of the same items). For example, one would be able to act out the rotation of a foot while watching its displacement.

Because users like to author cycles by performing them several times — each time refining the motion — we begin by extracting a single cyclic curve from the multiple performed cycles. Our solution is formulated as an optimization problem which takes a repetitive and nearly cyclic motion as input, and outputs a single clean cyclic motion (Section 4). This optimization is performed in the N-dimensional space, where N is the number of dimensions being captured — as shown in our examples, N can vary from 1 or 2 (mouse input, Fig. 8) to a few thousands (physical simulation, Fig. 10). The cycle extraction may be applied to any degree of freedom parameterizing the character's pose, such as rig controllers, skeletons, vertices, and any transformation such as translations, rotations, scales.

To allow the user to directly edit and refine the cyclic motion from a single viewport, we introduce in Section 5 *MoCurves*, a curve editor that combines translations, rotations, scalings and timing into a single unified geometric interface. We then describe in Section 5.3 an additional edit to our curves that allows specifying planar contacts and solving sliding effects with a single stroke.

4 CYCLE SPECIFICATION

In this section we describe how we automatically extract a single closed cycle from multiple *nearly cyclic* repetitions. The performed trajectory is a discrete function p which associates times t_i to a vector of values $\boldsymbol{\theta}_i = (\theta_i^1, \theta_i^2, ..., \theta_i^N)$. These values can represent any type of transformation (positions, orientations, scales, a mix of them, etc.). The performed trajectory p is supposed to be a periodic movement, but is imperfect in both space — noisy $\boldsymbol{\theta}_i$ — and time — noisy t_i. Hence our goal is to identify points p_i in the nearly cyclic motion that are geometrically similar across periods. We start by defining a curve descriptor that allows us to estimate the most probable value for the period T. We then extract a set of nearly cyclic curves that we average in order to obtain the shape of the final signal. We finally stitch the extremities together in order to obtain a perfectly looping cycle.

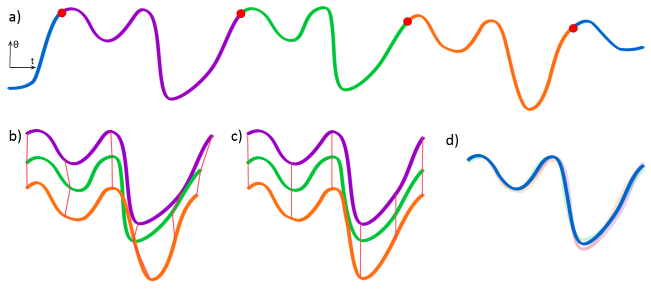

Figure 2: Steps of the cycle extraction algorithm illustrated on a simple example — only one attribute θ is animated, so p is of dimension 2. a) Points with a similar neighboring shape are identified (red circles) using a new curve descriptor. This allows to compute the average period of the motion, and to partition p into several cycles. b) Correspondences between cycles are computed. c) Each curve is non-uniformly scaled in order to align the corresponding points. d) These curves are averaged to form the final cycle.

Curve descriptor. The function p represents an $N+1$-dimensional curve, in which we seek to identify repetitive patterns. To measure the similarity between points in a curve, we devised a descriptor that characterizes the neighboring shape of a point on a curve. Our descriptor is similar to the one used by Mori et al. [Mori et al. 2005], which we extend to the n-dimensional case and make variant to rotations. Hence, we compute two descriptors $\boldsymbol{h}_i$ and $\boldsymbol{g}_i$ that respectively measure shape and velocity variations:

$$\boldsymbol{h}_i = \begin{bmatrix} \hat{\boldsymbol{h}}_i^0 \\ \dots \\ \hat{\boldsymbol{h}}_i^k \end{bmatrix} \quad where \;\; \hat{\boldsymbol{h}}_i^k = \sum_{|i-j|\le k} \boldsymbol{\theta}_j' \;\; and \;\; \boldsymbol{\theta}_j' = \frac{\boldsymbol{\theta}_{j+1} - \boldsymbol{\theta}_{j-1}}{t_{j+1} - t_{j-1}}$$

$$\boldsymbol{g}_i = \begin{bmatrix} \hat{\boldsymbol{g}}_i^0 \\ \dots \\ \hat{\boldsymbol{g}}_i^k \end{bmatrix} \quad where \;\; \hat{\boldsymbol{g}}_i^k = \sum_{|i-j|\le k} \boldsymbol{\theta}_j'' \;\; and \;\; \boldsymbol{\theta}_j'' = \frac{\boldsymbol{\theta}_{j+1}' - \boldsymbol{\theta}_{j-1}'}{t_{j+1} - t_{j-1}}$$

where we used $k = 5$, and we then define the similarity between two points as:

$$d_D(i,j) = \left\|\boldsymbol{h}_i - \boldsymbol{h}_j\right\|_2 \cdot \left\|\boldsymbol{g}_i - \boldsymbol{g}_j\right\|_2 .$$

Period evaluation. Considering the i-th point of the curve, p_i, we define J_i as the set of all indices j that are a local minima of $d_D(i,j)$, while remaining under a threshold d_D^m. By construction, J_i contains points that are similar in shape and speed to p_i. By conservatively choosing d_D^m high enough — thus favoring to select too many rather than too few points — we ensure that all the points corresponding to p_i in other cycles are present in the set J_i. Then, we eliminate the outliers in J_i and divide the period T by computing the minimal period that satisfies:

$$T = \min_{T \in \mathbb{N}} T \quad s.t. \quad \begin{array}{l} \forall k \in \{\left\lceil \frac{t_1 - t_i}{T} \right\rceil, \dots, \left\lfloor \frac{t_n - t_i}{T} \right\rfloor\}, \\ \exists j \in J_i \; s.t. \, |t_j - (t_i + kT)| < m_T \end{array}$$

Here, the threshold m_T depicts the variation in time of the cycles in p; we used $m_T = \lceil T/8 \rceil$, which we found reasonable in practice for cyclic motions performed by humans. In some particular cases, often when two parts of the motion are very similar in shape and velocity, the point i can be badly chosen resulting in an incorrect period. To eliminate this undesirable case, we perform this computation for ten random points and select the median period.

Average cycle. Given the set of indices J_i cleaned from outliers (i.e. J_i only contains points corresponding to p_i in other cycles), we cut the curve p at the corresponding points and extract a set of cycles $c_1, c_2, \dots, c_p$ (Fig. 2a). Similarly to the construction of J_i, we measure the similarity d_D over evenly spaced points to find correspondences between the cycles (Fig. 2b): $\boldsymbol{c}_1(t_1^k) \leftrightarrow \boldsymbol{c}_2(t_2^k) \leftrightarrow \dots \leftrightarrow \boldsymbol{c}_p(t_p^k)$. We then non-uniformly scale the cycles such that $t_i^k = t_j^k \; \forall i,j,k$ and the temporal length of each cycle is T (Fig. 2c). We finally compute the average cycle $\boldsymbol{c}$ (Fig. 2d) yielding:

$$\boldsymbol{c}(t) = \frac{1}{p} \sum_{i=1}^{p} \boldsymbol{c}_i(t), \;\; \forall t \in [0 \dots T]$$

The curve c we averaged may contain a discontinuity at its extremities, i.e. $\boldsymbol{d}_{ex} = \boldsymbol{c}(0) - \boldsymbol{c}(T)$ may not be $\boldsymbol{0}$. In order to make it perfectly cyclic, we stitch the curve c as follows:

$$\boldsymbol{c}(t) = \boldsymbol{c}(t) + \left(\frac{t}{T} - \frac{1}{2}\right) \boldsymbol{d}_{ex}, \;\; \forall t \in [0 \dots T]$$

Spline fitting. In order to have a smoother representation of the motion, as well as a simpler editing, we fit a cubic Bezier curve to each component of the cycle c — i.e. one for the translations (γ_P), one for the rotations (γ_R) and one for the scales (γ_S) of each moving item. Many B-Spline fitting methods already exist; we chose to use the one implemented in the Autodesk Maya API.

5 MOCURVES

To edit motion cycles in an intuitive manner, we combine spatial and temporal controls into a single geometric representation that allows the user to edit both aspects in a single viewport. This is particularly challenging for time as the user needs to precisely and intuitively edit temporal constraints using 3D spatial manipulations.

For this purpose, we introduce MoCurves. For each animated *item* (i.e. object, rig controller, bone, etc.), we define one MoCurve that represents its spatial (i.e. translation, rotation and scale) and temporal transformations. The following subsections describe how MoCurves allow the visualization and manipulation of diverse aspects of the motion, as well as permit the simple editing of planar contacts and automatic solving of sliding effects. Note that to amplify the intuition of movement and timing, we provide the ability to make all manipulations in real-time, while the animation is played; a benefit of working with motion cycles is that they seamlessly loop, so this does not create a visual discomfort.

5.1 Spatial manipulations

As described in Section 4, an item's position, rotation and scale over time are described by cubic Bezier splines γ_P, γ_R and γ_S. There is a direct correspondence between the parametrization of these curves and the time of the animation (e.g. at time t^*, the item is at position $\gamma_P(s_P^*)$). We note φ_P, φ_R and φ_S the bijective functions giving t from the parametrization of each curve — i.e. $t^* = \varphi_P(s_P^*) = \varphi_R(s_R^*) = \varphi_S(s_S^*)$. This provides a unified correspondence between all aspects of the motion.

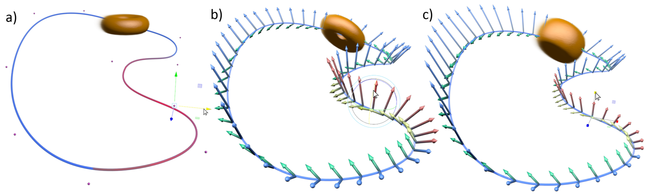

Figure 3: MoCurves allow three types of spatial edition. A 3D spline with editable control points represents the trajectory over time (a). Arrows at every time frame represent both the orientation (b) and scale (c) over time; they are directly manipulable in the viewport. In red are the regions affected by the manipulations.

The curve γ_P is displayed in the viewport in order to represent the displacement (i.e. translations) over time. The user can directly edit the trajectory by manipulating control points of γ_P (Fig. 3a). We represent the orientation (i.e. rotations) at each time frame by two orthogonal arrows centered at the corresponding position (Fig. 3b), and the scaling by the geometry of these same arrows (Fig. 3c).

Thus, by looking at a MoCurve, a user has a clear overview of the item's movement.

The orientation and scale of arrows can directly be manipulated in order to modify the rotations and scales of the curve over time. The modification of one frame also modifies neighboring ones in order to smooth the motion, just like the editing of a curve's control point affects a certain fraction of the curve. The range τ of affected frames (red arrows in Fig. 3) is set by the user. When an attribute of arrow i is deformed by $d\theta_i$, we propagate the deformation on arrows j using a Gaussian radial basis function:

$$d\theta_j = d\theta_i \cdot e^{-\sigma(i-j)^2}, \quad \forall j \in \{i-\tau, i+\tau\}$$

5.2 Temporal manipulations

Animators often draw time bars at a constant temporal interval to convey the timing of a hand-drawn motion. Inspired by this representation, we render on top of the Mocurve γ_P *keypoints* at each time frame of the animation. A keypoint can either be represented by a point or by arrows that also give information about orientation and scale (see Subsection 5.1). Hence, if keypoints are close to each other it means that the motion is slow, while if they are very distant it means that the motion is very fast.

Once again, the visual representation also serves as a manipulation tool: the user can directly edit the position of keypoints in order to edit the timing of the cycle, as shown in Fig. 4. We formulate the deformation as shape preserving deformation, similarly to [Kim et al. 2009], but here applied to the case of periodic curves.

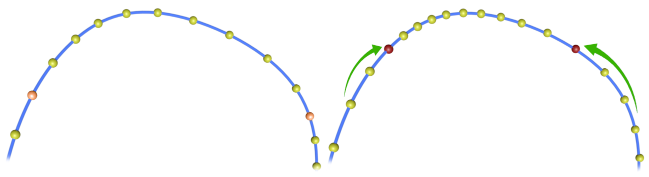
Figure 4: Two keypoints are moved upwards on the curve, setting new timing constraints (red points). A smooth time warping is applied, resulting in a motion that is slower at the top. Note that the motion is consequently faster on the rest of the curve in order to conserve the cycle period.

By moving keypoints, the user defines a set of spatial constraints: $\gamma_P(s_i^*) = \gamma_P(s_i^c), \forall i \in C$. We thus seek a new distribution of keypoints along the curve $s^* = (s_0^*, ..., s_m^*)$ satisfying $s_i^* = s_i^c, \forall i \in C$. This is equivalent to computing a new temporal distribution $t^* = (t_0^*, ..., t_m^*)$ satisfying $t_i^* = \varphi_P(s_i^c), \forall i \in C$. To do so, we solve a quadratic optimization problem containing four energy terms.

Constraints $E_C(t^*)$. We penalize the distance between the constrained points and their desired position:

$$E_C(t^*) = \sum_{i \in C} \left(t_i^* - \varphi_P(s_i^c)\right)^2$$

Period $E_T(t^*)$. A crucial constraint is that cycles must conserve the period:

$$E_T(t^*) = (t_m^* - t_0^* - T)^2$$

Velocity $E_V(t^*)$. We regularize velocities by penalizing deviations from the ones on the original curve. In other words, the temporal spacing between consecutive t_i^* should be stable:

$$E_V(t^*) = \sum_{i=0}^{m-1} \left((t_{i+1}^* - t_i^*) - (t_{i+1} - t_i)\right)^2$$

Speed variation $E_S(t^*)$. The time warping must not introduce points where the motion is accelerated or decelerated abruptly. That means that two consecutive segments have to stay close in size:

$$E_S(t^*) = \sum_{i=0}^{m-1} \left((t_{i+1}^* - t_i^*) - (t_i^* - t_{i-1}^*)\right)^2$$

The new time distribution is thus obtained by minimizing the total energy, while ensuring that t^* increases:

$$\begin{array}{ll} \min & w_1 E_C(t^*) + w_2 E_T(t^*) + w_3 E_V(t^*) + w_4 E_S(t^*) \\ \text{subject to} & (t_{i+1}^* - t_i^*) \geq 0, \ \forall i \in \{0, ..., m-1\} \end{array}$$

We solve this constrained quadratic programming problem using a qp solver, where we choose the weights w_1 and w_2 to be 10^4 times bigger than w_3 and w_4 because they act as strong constraints. Finally, we recover the positions, orientations and scales at each time step using $\gamma_P(\varphi_P^{-1}(t_i^*))$, $\gamma_R(\varphi_R^{-1}(t_i^*))$ and $\gamma_S(\varphi_S^{-1}(t_i^*))$.

5.3 Contacts

While contacts are present in most cyclic character motions such as locomotion, they can hardly be acted out accurately as they involve sharp corners in the trajectory. To edit a MoCurve as to exhibit sharp corners and straight lines, we introduce a stroke-based editor that cuts the curve with a straight line. This line, extruded along the viewing direction, defines a plane on which all points from a section of the curve are projected (Fig. 5).

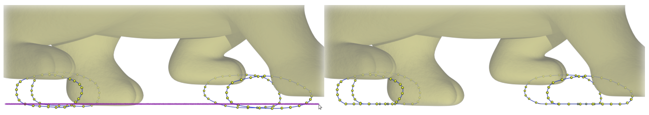
Figure 5: The user can draw a line (here in purple) to specify ground contacts. Portions of the trajectories are then projected on the contact plane, as shown on the right.

This solves contacts in terms of space, but not in terms of time. Indeed, two points simultaneously touching the ground can introduce a sliding effect if they are not coordinated. In other words, we need to ensure that at every frame of the animation, two items being in contact with the ground have the same velocity — note that velocities are expressed with respect to the character's root. The earlier projection step gives us information about which items are in contact and at which time. This allows us to construct a graph of the velocities of items in contact over time, as shown in Fig. 6. We fit a spline λ minimizing the distance to these curves, using a method similar to Section 4. λ defines the desired velocity of all items in contact with the ground over time.

We now spatially modify the curve γ_P of each colliding item in order to satisfy the contact velocity λ. First, we transform the part of γ_P being in contact, which we define as $\gamma_P^C \subset \gamma_P$, such that:

$$\gamma_P^C(\varphi^{-1}(t + dt)) = \gamma_P^C(\varphi^{-1}(t)) + \lambda(t) \cdot dt, \ \forall t$$

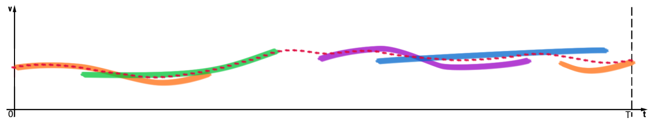

Figure 6: This graph represents the velocity over time of four items touching the ground (one per color). To unify them, we fit a spline λ (red dashed curve) that averages the contacts velocity. Note that here we illustrate for only one dimension, but the velocity is usually 3D.

This may change the length of γ_P^C by a scale factor s_l. In order to keep a smooth connection between γ_P^C and the rest of the curve, we also scale $\gamma_P \setminus \gamma_P^C$ by s_l in the direction of the contact. Finally, in order to ensure consistent contacts — i.e. contact points are fixed in world space — we move the root of the character with the velocity -$\lambda(t)$ in world space coordinates. This way, by simply specifying a contact surface, the user is able to make a character move inside the environment in a consistent way, without sliding effect.

6 RESULTS

We apply our system to four different characters with diverse shapes and rig complexities. In this section, we present a number of compelling motion cycles that both novice and expert users were able to author using a variety of input devices. Table 1 gives statistics about the number of elements and transformations that were animated in each case — note that diverse full-body motion capture suits were used for the Mocap motions of Fig. 9, which is why the numbers vary at the bottom of the table. All the resulting animations, as well as some steps of the authoring process, are shown in the accompanying video.

Motion cycle	Figure	Anim. elem.	Anim. transf.
Dinosaur walk	7 (a)	13	39
Human punch	7 (b)	5	8
Robot swim	7 (c)	8	23
Human dance	7 (d)	12	33
Dinosaur dance	7 (e)	18	49
Dragon eat	8 (left)	10	29
Dragon fly	8 (middle)	35	51
Human juggle	8 (right)	16	52
Dinosaur leap	9 (left)	11	21
Human kick	9 (middle)	11	30
Mocap punch	9 (right)	21	126
Mocap walk	9 (right)	60	253
Mocap samba	9 (right)	52	159

Table 1: For each motion cycle presented in the Results section, this table gives the number of elements (i.e. rig controllers or skeleton joints) and the number of transformations (i.e. translations, rotations, scales) that were animated.

To evaluate the accessibility of our system, we invited five novice users, who never animated any character before, and gave them a limit of one hour to author a motion cycle using our tool. The resulting animations, presented in Fig. 7 and in the accompanying video, are particularly convincing considering the inexperience of the creators. This study was also a social success as users were enchanted to be able to animate a character, several of them concluding: "this was the most enjoyable user study of my life". For comparison, we also asked two of these novice users to create a similar motion cycle using Autodesk Maya, without our plugin. After one hour of struggle (which was their time limit), no exploitable content was created. This result confirms our belief that general purpose animation packages such as Maya require significant training before even simple animations can be created.

Additionally, our tool was used by a professional artist in order to evaluate how well our system is integrable into an expert's pipeline. The artist used a Wacom pen and tablet and authored the three motion cycles presented in Fig. 8: the eating dragon was animated in approximately 20 minutes, the flying dragon in approximately 15 minutes and the juggling human in approximately 35 minutes. The last example exhibits how much our system enables a fine control over the spatial and temporal aspects of the motion, allowing the composition of complex synchronizations. As a feedback, the artist shared how delighted he was to be able to directly perform the motion he had in mind without having to be super precise, and how important the MoCurves were in order to maintain a full control over the final result. He claimed that this is a tool he would like to use regularly for the motion cycles he needs to create.

A large variety of devices can be used to practice performance animation: these range from a computer mouse to full body motion-capture suits and include Leap Motion, Kinect [Wang et al. 2012], tactile surfaces [Chung et al. 2015; Lockwood and Singh 2012] and other dedicated devices [Glauser et al. 2016; Oore et al. 2002; Shiratori et al. 2013; Slyper and Hodgins 2008]. Our method is generic enough to work with any type and dimensionality of data as input. Most of our results were generated using the most familiar devices — a mouse or a digital pen — but we also demonstrate in Fig. 9 and in the accompanying video proper functioning with Leap Motion, HTC Vive and full body motion-capture suits.

In terms of performances, the system is responsive enough to permit an interactive utilization. The cycle extraction algorithm takes less than one second to be executed, even in high-dimensional cases (such as motion-capture, Fig. 9) where the curve being analyzed can have several hundreds of dimensions. As for the MoCurves manipulations, as specified in Section 5, they are executed in real-time, while the animation is being played.

7 LIMITATIONS AND FUTURE WORK

When extracting a cycle from a performed motion, our algorithm requires rough consistency in the input cycles, both in shape (same overall displacement) and timing (cycles of similar duration). If the recorded performance cycles vary widely, the algorithm will fail to find a period or the extracted loop will be of poor quality. However, in our experiments, even inexperienced users were able to perform loops consistent enough to deliver quality results.

MoCurves allow the visualization and editing of the most widely used attributes in animation. However, artists sometimes customize their rigs with additional attributes, such as roll and lean for a foot. In our results, we supplanted them with the translation or rotation of additional rig elements, but it would be interesting to explore

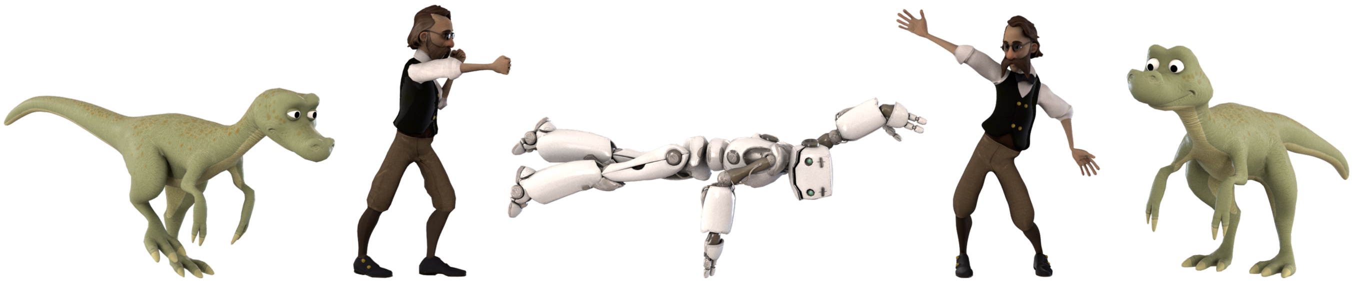

Figure 7: Five novice users, who never created a character animation before, used our system. In less than one hour, they were respectively able to create these motion cycles. From left to right: a walk, a punch, a swim and two dance cycles.

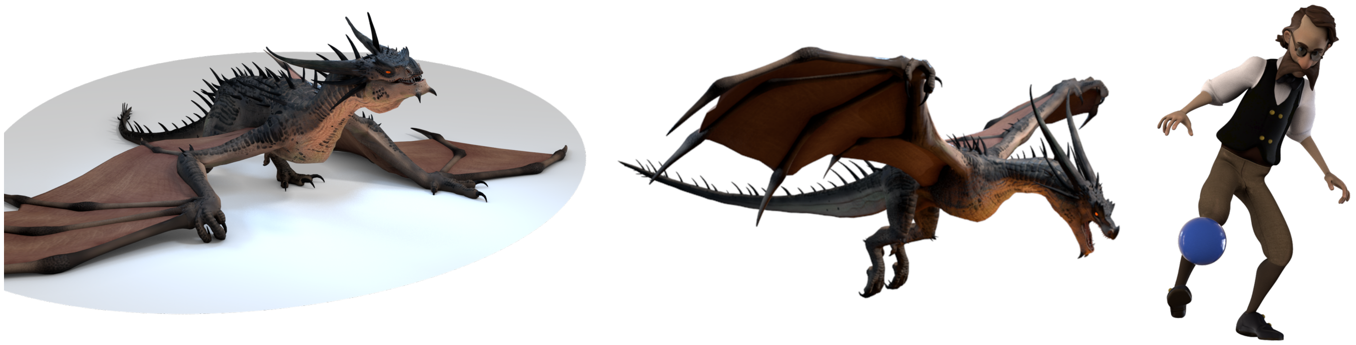

Figure 8: A professional artist used our system to author these three motion cycles. From left to right: a dragon eating, a dragon flying and a human juggling a ball. They were respectively created in about 20 minutes, 15 minutes and 35 minutes.

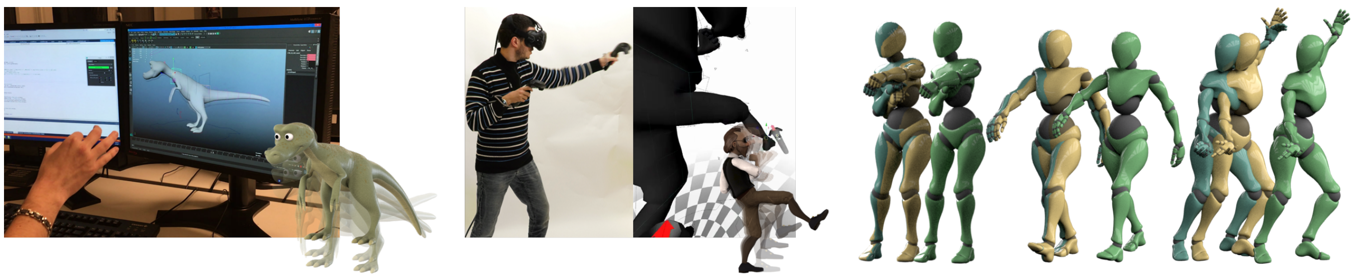

Figure 9: Our system supports a variety of performance capture devices. Here we show a jump cycle created using the Leap Motion hand tracker (left), a kick cycle created using the HTC Vive (middle), and a punch, a walk and a samba cycles created using a full body motion-capture suit (right). In these last examples, the overlapped blue and yellow mannequins show the spatial difference of cycles in the imperfect performed motion, while the green mannequin shows the looping cycle extracted by our algorithm. Note that in each case, we solve for a single multidimensional curve representing the whole motion (dimension 127 for the punch cycle, 254 for the walk cycle and 160 for the samba cycle).

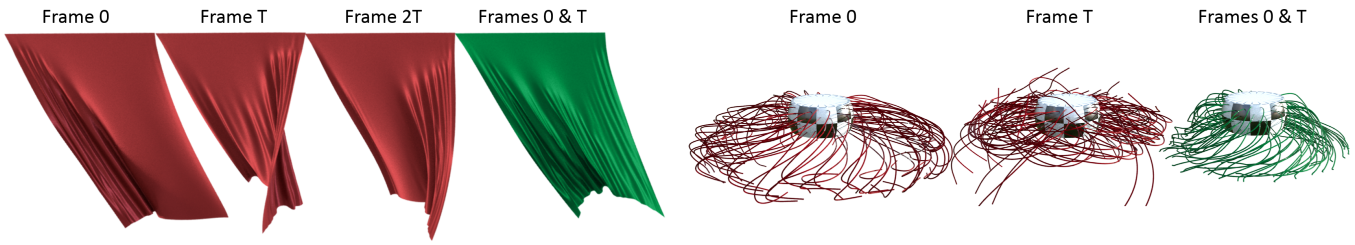

Figure 10: Handles of these two physics simulations (i.e. translating top of cloth and rotating basis of hair) move periodically. Though, the simulated motions are not periodic, as shown in red with frames spaced by the period T. Our algorithm extracts looping motions, in green, from these simulations. The dimensionality of the cyclified curves is respectively 7804 and 2611.

mechanisms that would enhance MoCurves for the representation and manipulation of supplementary attributes. As well, we provide a way to easily edit planar contacts, but the intuitive authoring of more complex interactions, such as non-planar or dynamic contacts, remains an open and challenging problem.

Our algorithm can find cycles even in structured physical simulations such as cloth, as shown in Fig. 10. However, it does not conserve the physical correctness of the original simulation. In the future, we would like to integrate mechanical constraints into our algorithm in order to create physically-accurate cyclic simulations.

8 CONCLUSION

In this work, we introduced an authoring tool tailored to cyclic motion design. Our tool enables a user to repeatedly act out a periodic movement and automatically extract a single closed cyclic motion. In order to conserve a fine level of control required by artists, we then introduced MoCurves, a curve editor that combines both space and time into a single geometric entity that allows coordinated editing in a single viewport. By removing a thick layer of expert knowledge required by general purpose animation tools, we allowed both professional artists and novice users to create compelling animations.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 642841. We would like to thank the participants of our user study for their time and the fruitfull disussions.

REFERENCES

Amr Ahmed, Farzin Mokhtarian, and Adrian Hilton. 2003. Cyclification of Human Motion for Animation Synthesis. In *Eurographics 2003 - Short Presentations.*

Jinxiang Chai and Jessica K. Hodgins. 2005. Performance Animation from Low-dimensional Control Signals. *ACM Trans. Graph.* 24, 3 (2005), 686–696.

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 (2016), 146:1–146:12.

Byungkuk Choi, Mi You, and Junyong Noh. 2008. Extended Spatial Keyframing for Complex Character Animation. *Comput. Animat. Virtual Worlds* 19, 3-4 (2008), 175–188.

M. G. Choi, K. Yang, T. Igarashi, J. Mitani, and J. Lee. 2012. Retrieval and Visualization of Human Motion Data via Stick Figures. *Comput. Graph. Forum* 31, 7pt1 (2012), 2057–2065.

Se-Joon Chung, Junggon Kim, Shangchen Han, and Nancy S. Pollard. 2015. Quadratic Encoding for Hand Pose Reconstruction from Multi-Touch Input. In *EG 2015 - Short Papers.*

Loïc Ciccone, Martin Guay, and Robert W. Sumner. 2016. Flow Curves: an Intuitive Interface for Coherent Scene Deformation. *Computer Graphics Forum* 35, 7 (2016), 247–256.

Patrick Coleman, Jacobo Bibliowicz, Karan Singh, and Michael Gleicher. 2008. Staggered Poses: A Character Motion Representation for Detail-preserving Editing of Pose and Coordinated Timing. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 137–146.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Trans. Graph.* 29, 4 (2010), Article 130.

Sean Curtis, Ming C. Lin, and Dinesh Manocha. 2011. Walk This Way: A Lightweight, Data-Driven Walking Synthesis Algorithm. In *MIG.*

Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered Acting for Character Animation. *ACM Trans. Graph.* 22, 3 (2003), 409–416.

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 (2016).

Martin Guay, Marie-Paule Cani, and Rémi Ronfard. 2013. The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph.* 32, 6 (2013), 205:1–205:8.

Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time Sketching of Character Animation. *ACM Trans. Graph.* 34, 4 (2015), 118:1–118:10.

Fabian Hahn, Frederik Mutzel, Stelian Coros, Bernhard Thomaszewski, Maurizio Nitti, Markus Gross, and Robert W. Sumner. 2015. Sketch Abstractions for Character Posing. In *Proc. of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation.* 185–191.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proc. of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques.* 71–78.

Eugene Hsu, Marco da Silva, and Jovan Popović. 2007. Guided Time Warping for Motion Editing. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 45–52.

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. Spatial Keyframing for Performance-driven Animation. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 107–115.

Jaewoong Jeon, Hyunho Jang, Soon-Bum Lim, and Yoon-Chul Choy. 2010. A sketch interface to empower novices to create 3D animations. *Computer Animation and Virtual Worlds* 21, 3-4 (2010), 423–432.

Ming Jin, Dan Gopstein, Yotam Gingold, and Andrew Nealen. 2015. AniMesh: Interleaved Animation, Modeling, and Editing. *ACM Trans. Graph.* 34, 6 (2015), 207:1–207:8.

Jongmin Kim, Yeongho Seol, and Jehee Lee. 2013. Human motion reconstruction from sparse 3D motion sensors using kernel CCA-based regression. *Computer Animation and Virtual Worlds* 24, 6 (2013), 565–576.

Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. 2009. Synchronized Multi-character Motion Editing. In *ACM SIGGRAPH 2009 Papers.* 79:1–79:9.

Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. 2000. Interactive Control for Physically-based Animation. In *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques.* 201–208.

Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. 2002. Interactive Control of Avatars Animated with Human Motion Data. In *Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques.* 491–500.

Juncong Lin, Takeo Igarashi, Jun Mitani, and Greg Saul. 2010. A Sketching Interface for Sitting-pose Design. In *Proc. of the Seventh Sketch-Based Interfaces and Modeling Symposium.* 111–118.

Huajun Liu, Xiaolin Wei, Jinxiang Chai, Inwoo Ha, and Taehyun Rhee. 2011. Realtime Human Motion Control with a Small Number of Inertial Sensors. In *Symposium on Interactive 3D Graphics and Games.* 133–140.

Noah Lockwood and Karan Singh. 2011. Biomechanically-inspired Motion Path Editing. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 267–276.

Noah Lockwood and Karan Singh. 2012. Finger Walking: Motion Editing with Contact-based Hand Performance. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 43–52.

Tyler Martin and Michael Neff. 2012. Interactive Quadruped Animation. In *MIG.*

Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. 2009. Interactive Generation of Human Animation with Deformable Motion Models. *ACM Trans. Graph.* 29, 1 (2009), 9:1–9:12.

Greg Mori, Serge Belongie, and Jitendra Malik. 2005. Efficient Shape Matching Using Shape Contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 11 (2005), 1832–1837.

Tomohiko Mukai. 2011. Motion Rings for Interactive Gait Synthesis. In *Symposium on Interactive 3D Graphics and Games.* 125–132.

Tomohiko Mukai and Shigeru Kuriyama. 2009. Pose-timeline for Propagating Motion Edits. In *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 113–122.

Michael Neff, Irene Albrecht, and Hans-Peter Seidel. 2007. Layered Performance Animation with Correlation Maps. *Comput. Graph. Forum* 26 (2007), 675–684.

Sageev Oore, Demetri Terzopoulos, and Geoffrey Hinton. 2002. A Desktop Input Device and Interface for Interactive 3D Character Animation. In *In Proc. Graphics Interface.* 133–140.

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. of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 155–164.

Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. 1996. Efficient Generation of Motion Transitions Using Spacetime Constraints. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques.* 147–154.

Takaaki Shiratori, Moshe Mahler, Warren Trezevant, and Jessica K. Hodgins. 2013. Expressing animated performances through puppeteering. In *2013 IEEE Symposium on 3D User Interfaces (3DUI).* 59–66.

Fernando Wagner da Silva, Luiz Velho, Jonas Gomes, and Siome Goldenstein. 1999. Motion Cyclification by Time x Frequency Warping. In *Proceedings of the XII Brazilian Symposium on Computer Graphics and Image Processing.* 49–.

Ronit Slyper and Jessica K. Hodgins. 2008. Action Capture with Accelerometers. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 193–199.

Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. 2011. Motion Reconstruction Using Sparse Accelerometer Data. *ACM Trans. Graph.* 30, 3 (2011), 18:1–18:12.

Sílvio César Lizana Terra and Ronald Anthony Metoyer. 2007. A performance-based technique for timing keyframe animations. *Graphical Models* 69, 2 (2007), 89–105.

Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion Doodles: An Interface for Sketching Character Motion. In *ACM SIGGRAPH 2004 Papers*. 424–431.

Benjamin Walther-Franks, Marc Herrlich, Thorsten Karrer, Moritz Wittenhagen, Roland Schröder-Kroll, Rainer Malaka, and Jan Borchers. 2012. Dragimation: Direct Manipulation Keyframe Timing for Performance-based Animation. In *Proc. of Graphics Interface 2012*. 101–108.

Xin Wang, Qing Ma, and Wanliang Wang. 2012. Kinect driven 3D character animation using semantical skeleton. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, Vol. 01. 159–163.

Xiaolin Wei and Jinxiang Chai. 2011. Intuitive Interactive Human-Character Posing with Millions of Example Poses. *IEEE Comput. Graph. Appl.* 31, 4 (2011), 78–88.

Andrew Witkin and Zoran Popovic. 1995. Motion Warping. In *Proc. of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*. 105–108.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. In *ACM SIGGRAPH 2007 Papers*.

Innfarn Yoo, Michel Abdul Massih, Illia Ziamtsov, Raymond Hassan, and Bedrich Benes. 2015. Motion Retiming by Using Bilateral Time Control Surfaces. *Comput. Graph.* 47, C (2015), 59–67.

Innfarn Yoo, Juraj Vanek, Maria Nizovtseva, Nicoletta Adamo-Villani, and Bedrich Benes. 2014. Sketching Human Character Animations by Composing Sequences from Large Motion Database. *Vis. Comput.* 30, 2 (2014), 213–227.