SMEAR: Stylized Motion Exaggeration with ARt-direction

Jean Basset, Pierre Bénard, Pascal Barla

SMEAR: Stylized Motion Exaggeration with ARt-direction

JEAN BASSET, PIERRE BÉNARD, and PASCAL BARLA,
Inria, Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, France

Fig. 1. We present a method for the stylization of 3D motion using smear frames. From left to right on the three central frames, we show elongated in-betweens, motion lines and multiple in-betweens. All three stylization effects are based on automatically-computed motion offsets, shown color-coded in inset figures. These effects also depend on vertex velocity in this case, to limit the stylization to fast moving parts only.

Smear frames are routinely used by artists for the expressive depiction of motion in animations. In this paper, we present an automatic, yet art-directable method for the generation of smear frames in 3D, with a focus on elongated in-betweens where an object is stretched along its trajectory. It takes as input a key-framed animation of a 3D mesh, and outputs a deformed version of this mesh for each frame of the animation, while providing for artistic refinement at the end of the animation process and prior to rendering.

Our approach works in two steps. We first compute spatially and temporally coherent *motion offsets* that describe to which extent parts of the input mesh should be leading in front or trailing behind. We then describe a framework to stylize these motion offsets in order to produce elongated in-betweens at interactive rates, which we extend to the other two common smear frame effects: multiple in-betweens and motion lines. Novice users may rely on preset stylization functions for fast and easy prototyping, while more complex custom-made stylization functions may be designed by experienced artists through our geometry node implementation in Blender.

CCS Concepts: • **Computing methodologies** → **Non-photorealistic rendering**; **Animation**.

Additional Key Words and Phrases: Stylized Animation, Smear Frames

ACM Reference Format:
Jean Basset, Pierre Bénard, and Pascal Barla. 2024. SMEAR: Stylized Motion Exaggeration with ARt-direction. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641519.3657457

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

1 INTRODUCTION

Stylization has long been applied to traditional hand-drawn 2D animation to expressively convey motion. The 12 principles of animation [Thomas and Johnston 1981] developed at the Walt Disney Studio in the late 1920's and early 1930's have served as guidelines since then. Lasseter [1987] presents how these principles can be manually applied by an artist to 3D animations. In particular, he describes how stretching an object to cover the span of several frames helps relieve the disturbing effect of strobing that happens with fast motion. This technique precisely corresponds to *elongated in-betweens* that are typical of 2D animations [Williams 2001], as illustrated in Figure 2. Drawing *multiple in-betweens*, i.e., copies of the animated objects displaced along their trajectories, is an alternative smearing technique that is used to ease motion perception. Another common motion stylization effect, inspired by comic books, is *motion lines* that clearly emphasize motion trajectories. With the recent popularity of 3D animated films with strong stylistic identities, such as the "Spider-Verse" movies, these motion effects have regained interest. In this work, we aim at recreating smear frames, especially elongated in-betweens, in 3D animations with a semi-automatic, art-directed approach.

To guide the design of our method and better understand how smear frames are used in 3D productions, we conducted an interview

Fig. 2. Example of smear frames in traditional 2D animation, "The Dover Boys at Pimento Academy", directed by Charles M. Jones (public domain).

with the head of innovation in an animation studio (see the supplemental document for details) and had more informal discussions with professional animators. Combined with observational studies of traditional 2D animations and tutorials on smear frames [Williams 2001], we have identified the following guidelines:

G1: Smear frames have two main goals: (1) increase retinal persistence by creating a larger overlap among consecutive frames, and (2) create an effect of surprise (the "snap!" in Richard Williams [2001] words) to direct the observer's gaze.

G2: A motion stylization tool must seamlessly fit in the 3D animation workflow. It must thus be fast enough to provide interactive feedback, toggling the motion effect should be easy, and artists must be able to fine tune the results manually. Moreover, for skeletal animations, the tool should not modify the input poses.

G3: Motion stylization effects should be fully customizable by artists, allowing to create unique styles for every production.

As discussed in Section 2, no existing method proposed in academic work fully satisfies these guidelines. First, smear frames, as described in **G1**, have received little attention compared to more generic *squash-and-stretch* deformations, and no method specifically addressed elongated in-betweens. Second, existing methods targetting other motion effects or smear frame styles are difficult to integrate into the animation workflow (**G2**) since they either involve simulations that offer little intuitive artistic control (e.g., [Zhang et al. 2020]) or are applied at the compositing stage (e.g., [Schmid et al. 2010]), which makes results hard to control by animators. In 3D animation software, such as Autodesk 3Dsmax or Blender, some smear frame creation tools are available as plugins, e.g., OverMorpher and motionFX. However, the former requires manual sculpting of the 3D model, which makes iterative refinement of the effect very tedious (**G2**); whereas the latter offers very little artistic control (**G3**).

In this paper, we present for the first time a method for the automatic yet art-directable creation of elongated in-betweens for keyframe-based animations of 3D meshes, which we extend to the cases of multiple in-betweens and motion lines. All three motion stylizations are based on *motion offsets*, which identify which parts of an input object or articulated figure should be leading in front or trailing behind once stylized. As detailed in Section 3, motion offsets are spatially- and temporally-coherent signed displacements of mesh vertices computed from their trajectories.

We then propose in Section 4 a framework that provides interactive artistic control over the stretching of an animated mesh along its trajectory based on motion offsets, to create elongated in-betweens that overlap adjacent frames (**G1**). The style of the smear frames can be controlled using predefined stylization functions based on high-level parameters, while the output per-frame meshes have the same topology than the input and may still be manually fine-tuned by artists (**G2**). In addition, we demonstrate that, with little adjustments, this framework generates other types of smear frames, such as multiple in-betweens and motion lines.

We finally show in Section 5 a wide range of stylization effects, and demonstrate that our implementation as a Blender plugin manages to quickly (i.e., in a few seconds) recompute motions offsets after any change in the input animation, achieving interactive stylization performance and iterative refinement of the animation and motion effects (**G2**). We rely on the geometric node graph interface in Blender to let experienced artists create new stylizations (**G3**). Our Blender implementation is included in supplemental material and is openly available at https://github.com/MoStyle/SMEAR.

2 RELATED WORK

Since its inception, one of the goal of computer graphics has been to reproduce the expressiveness of traditional hand-drawn 2D animations. Many works have thus explored ways to (semi-)automatically deform objects or characters to reproduce effects typical of traditional 2D animations, such as squash-and-stretch, or anticipation and follow-through.

Physically-based deformations. A common approach to generate elastic deformations is to rely on physical simulations. For instance, custom physically-based models were developed to produce exaggerated cartoon effects [Chenney et al. 2002; Coros et al. 2012; Garcia et al. 2007; Müller et al. 2005] or jiggling on characters [Iwamoto et al. 2015; Rumman and Fratarcangeli 2014].

The more generic "Complementary Dynamics" framework of Zhang et al. [2020] creates secondary deformations on top of a skeleton-based animation by computing these in the subspace orthogonal to the rig. Even though real-time performance can be achieved [Benchekroun et al. 2023], physically-based deformations are limited in terms of art-directability (**G2** & **G3**) since artists only control simulation parameters, requiring trials and errors.

Kinematic approaches. Another line of work has thus explored simpler, direct kinematic approaches to reproduce cartoon effects. The time-based filter of Wang et al. [2006] applied to vertex trajectories can express anticipation and follow-through. For character animations, deformations of the bones themselves [Kwon and Lee 2012] can reproduce squash-and-stretch effects, but they are in contradiction with **G2** as they automatically alter the skeleton pose.

Geometric deformers based on bone velocities [Noble and Tang 2007; Rohmer et al. 2021] and accelerations [Kalyanasundaram et al. 2022] are able to reproduce a wide range of motion exaggeration effects, and offer artist control through painted weights and handles. Example-based methods [Dvorožňák et al. 2017; Roberts and Mallett 2013] provide even more natural and intuitive controls, but they must be set up manually for each specific animation. Most importantly, they are not adapted to produce smear frames (**G1**).

Specifically targeting novice users, Ma et al. [Ma et al. 2022] present a user-interface to apply motion stylization presets on top of a 3D animation. Their system allows fast prototyping, in line with guideline **G2**, and additionally offers a node-based interface to more experienced users for customizing effects, in accordance with **G3**. However, they only handle simple rigid animations.

Smear frames. Far fewer previous methods explicitely aim at generating smear frames. Schmid et al. [2010] propose a custom rendering engine that supports programmable motion effects. They manage to produce smear frames in various styles: motion blur, multiple in-betweens and motion lines, inspired by the motion depiction taxonomy of Cutting [2002]. Style customization (**G3**) is possible by construction, but reserved to experienced programmers. More

importantly, the artist cannot manually fine tune the results since they are produced at rendering time, in contradiction with **G2**.

Swept volumes, i.e., the union of all positions spanned by an object over the course of an animation, are used to convey 3D motion in static images and sculptures [Kazi et al. 2016; Zhang et al. 2018]. They may also be used to create smear effects close to elongated in-betweens by sweeping an object along its trajectory, as stated by guideline **G1**, such as motion trails [Sellán et al. 2021] or motion blur [Jones and Keyser 2005]. Even though swept volume methods produce meshes that may be fine tuned by artists (**G2**), they are not guaranteed to be homeomorphic to the input object, hiding high-frequency surface details and concavities, thus impeding the recognition of the object. In addition, their high computation time makes them impractical for interactive stylization (**G2**).

Our method produces smear frame stylization (**G1**) directly in 3D at interactive frame rates, seamlessly fitting in the 3D animator workflow and supporting artistic fine tuning in post-process (**G2**). It is thus more similar to swept volume approaches, while addressing their limitations through a method based on motion offsets. Our method could be used in complement to other motion effects such as physical deformations or kinematics approaches, in order to create compelling animations with secondary motions and smearing. Our approach additionally borrows the concept of a multi-level authoring tool from [Ma et al. 2022], whereby stylization presets are exposed with default parameters, and complemented by a node graph interface for advanced customization (**G3**).

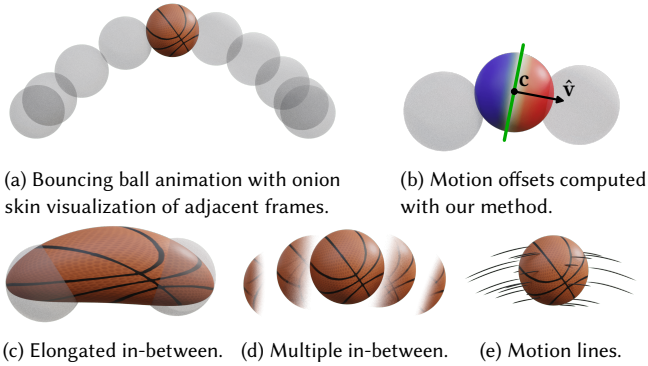
3 MOTION OFFSETS

In our approach, elongated in-betweens, multiple in-betweens or motion lines are all created at a given frame by first identifying which parts of the object should be leading in front, and which parts should be trailing behind (see Figure 3). Formally, we assign scalar values $\bar{\delta}_i(f) \in [-1, 1]$ at each vertex i and frame f, which we call *motion offsets* (color-coded in Figure 3b). Vertices with a motion offset of $\bar{\delta}_i(f) = 1$ (resp -1) will be the ones with the largest amount of stylization when depicting motion in the future (resp. in the past). For instance, in the case of elongated in-betweens, these points will be displaced so as to overlap the next (resp. previous) frame, as illustrated in Figure 3c. Such a normalization of motion offsets eases further stylization as it makes it more predictable.

In the following, we explain how we compute motion offsets, first in the case of a simple object (Section 3.1), then in the case of an articulated figure (Section 3.2). For clarity of notation, we usually omit the dependency of motion offsets on the current frame f. In addition, we write unit vectors as $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$.

3.1 Simple Object

For a globally-convex rigid object, such as the ball in Figure 3, motion offsets could be computed by a mere dot product between the normalized velocity of its centroid and the normal at a given vertex [Jones and Keyser 2005]. However, such a simple local formula is not adapted to objects with large concavities or high-frequency surface details, as motion offsets would exhibit undesired local variations. As shown in Figure 4-left, this can result in strong artifacts after stylization, such as important self-intersections, and hide concavities and surface details.



(a) Bouncing ball animation with onion skin visualization of adjacent frames.

(b) Motion offsets computed with our method.

(c) Elongated in-between. (d) Multiple in-between. (e) Motion lines.

Fig. 3. Motion offsets, separation plane (in green), and corresponding stylizations for a bouncing ball. Positive (resp. negative) motion offsets are painted red (resp. blue).

Fig. 4. Motion offsets and corresponding elongated in-between for a peanut model translating left to right, computed (left) as the dot product of the centroid velocity and vertex normals, and (right) with our approach.

We instead opt for a global characterization: we separate surface points leading in front from those trailing behind via a plane of normal $\hat{\mathbf{v}}$ – the normalized velocity of the object (i.e., its direction of motion) – and going through the object centroid $\mathbf{c}$ (see Figure 3b). The motion offset for a vertex i at position $\mathbf{p}_i$ is then given by the normalized signed distance to that plane:

$$\bar{\delta}_i = \frac{\delta_i}{\max_j |\delta_j|} \quad \text{with} \quad \delta_i = (\mathbf{p}_i - \mathbf{c}) \cdot \hat{\mathbf{v}}, \tag{1}$$

where the maximum is taken over all the vertices of the object.

For animations with fast changing motion directions, motion offsets may vary abruptly across frames at some vertices. We thus optionally smooth their values using a temporal window of N frames on either sides of the current frame f:

$$\bar{\delta}_i(f) = \sum_{n=-N}^{N} w_n \bar{\delta}_i(f+n), \tag{2}$$

with w_n a temporal weighting function. We use $w_n = \left[1 - \left(\frac{n}{N+1}\right)^2\right]^2$ with $N = 2$ in practice. The supplemental video shows the effect of this temporal smoothing on a fast animation.

3.2 Articulated figure

Compared to a simple object, an articulated figure exhibits more complex movements (due to different motions of skeleton bones), which must be captured by motion offsets. In this case, instead of separating mesh vertices with a plane as in Section 3.1, we separate them using ribbons that run through the bones and are locally

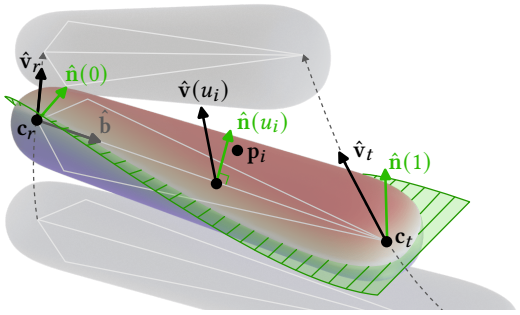


Fig. 5. Mathematical notations for a single bone. For every vertex i of the mesh with position $\mathbf{p}_i$, we compute the normal $\hat{\mathbf{n}}(u_i)$ of the ribbon (in green) which is orthogonal to the bone axis $\mathbf{b} = (\mathbf{c}_t - \mathbf{c}_r)$ and is as aligned as possible to the direction of motion $\hat{\mathbf{v}}(u_i)$ interpolated from the normalized velocities $\hat{\mathbf{v}}_r$ and $\hat{\mathbf{v}}_t$ at root and tip locations $\mathbf{c}_r$ and $\mathbf{c}_t$.

oriented as orthogonal as possible to the direction of motion, as illustrated in Figures 5 and 6. The rationale behind this choice is that we want to convey motion (**G1**) without changing the apparent skeleton pose (**G2**). We first explain how motion offsets are computed from ribbons for a single bone, and then for a full skeleton.

Single Bone. The geometry for a single bone is illustrated in Figure 5: the root and tip of a bone are denoted by $\mathbf{c}_r$ and $\mathbf{c}_t$ respectively, while its axis is written $\hat{\mathbf{b}}$ with $\mathbf{b} = \mathbf{c}_t - \mathbf{c}_r$. A vertex i with position $\mathbf{p}_i$ is mapped to the bone axis by assigning it a 1D parameter:

$$u_i = S\left(\frac{(\mathbf{p}_i - \mathbf{c}_r)\cdot\hat{\mathbf{b}}}{\|\mathbf{b}\|}\right), \tag{3}$$

with $S : \mathbb{R} \to [0,1]$ a smoothstep function classically defined by $S(x) = \left[3x^2 - 2x^3\right]_0^1$ where $[x]_0^1$ clamps x to the $[0,1]$ range.

We then compute a local direction of motion $\hat{\mathbf{v}}(u_i)$ through the spherical linear interpolation [Shoemake 1985] of motion directions at root and tip locations – noted $\hat{\mathbf{v}}_r$ and $\hat{\mathbf{v}}_t$ respectively – using:

$$\hat{\mathbf{v}}(u) = \frac{\sin((1-u)\omega)}{\sin(\omega)}\hat{\mathbf{v}}_r + \frac{\sin(u\omega)}{\sin(\omega)}\hat{\mathbf{v}}_t, \tag{4}$$

with $\omega = \arccos(\hat{\mathbf{v}}_r \cdot \hat{\mathbf{v}}_t)$. To avoid numerical inaccuracies, we fall back to a linear interpolation when $\omega < 0.1$ radians.

We next compute the ribbon normal $\hat{\mathbf{n}}(u_i)$ for vertex i, which should be orthogonal to the bone axis $\hat{\mathbf{b}}$, and aligned with the local direction of motion $\hat{\mathbf{v}}(u_i)$ as much as possible. The ribbon normal $\hat{\mathbf{n}}$ for a parametric position u along the bone axis is given by:

$$\mathbf{n}(u) = \hat{\mathbf{v}}(u) - \left(\hat{\mathbf{v}}(u)\cdot\hat{\mathbf{b}}\right)\hat{\mathbf{b}}, \quad \hat{\mathbf{n}}(u) = \frac{\mathbf{n}(u)}{\|\mathbf{n}(u)\|}. \tag{5}$$

The motion offset for vertex i is then given similarly to Equation 1:

$$\bar{\delta}_i = w_{\text{coll.}}(u_i)\frac{\delta_i}{\max_j |\delta_j|} \quad \text{with} \quad \delta_i = (\mathbf{p}_i - \mathbf{c}_r)\cdot\hat{\mathbf{n}}(u_i), \tag{6}$$

where $w_{\text{coll.}}(u) = 1 - (\hat{\mathbf{v}}(u)\cdot\hat{\mathbf{b}})^2$ is a weight function that decreases the magnitude of motion offsets as local motion directions get increasingly collinear to the bone axis. Its effect is best described in the case of multiple bones, as explained next.

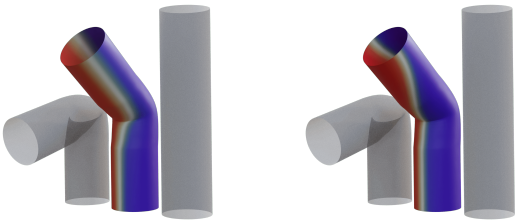
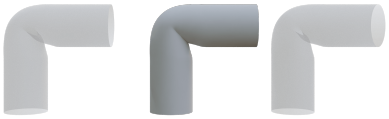
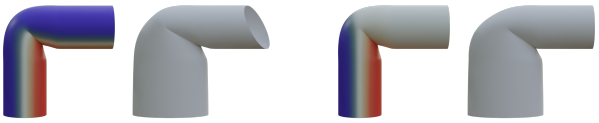


Fig. 6. Motion offsets for a two-bones capsule, with onion skin visualization of the animation. Left: motion offsets computed using a secant plane as in Section 3.1. Right: motion offsets computed using a ribbon separation surface depending on the local motion direction.



(a) Two bones capsule translating to the right with onion skin visualization.



(b) Without collinear weights. (c) With collinear weights.

Fig. 7. Motion offsets computed for the animation in (a), weighted according to motion collinearity (c), and without this weighting scheme (b), yielding a noticeable shearing.

Full skeleton. For a vertex i affected by multiple bones, its motion offset is computed by the contribution of each bone k through its skinning weight w_{ik}, similarly to [Rohmer et al. 2021]:

$$\bar{\delta}_i = \sum_k w_{ik}\,\bar{\delta}_{ik} \quad \text{with} \quad \sum_k w_{ik} = 1. \tag{7}$$

The motion offset contribution of a bone in a full skeleton is similar to that of a single bone but differs in term of normalization:

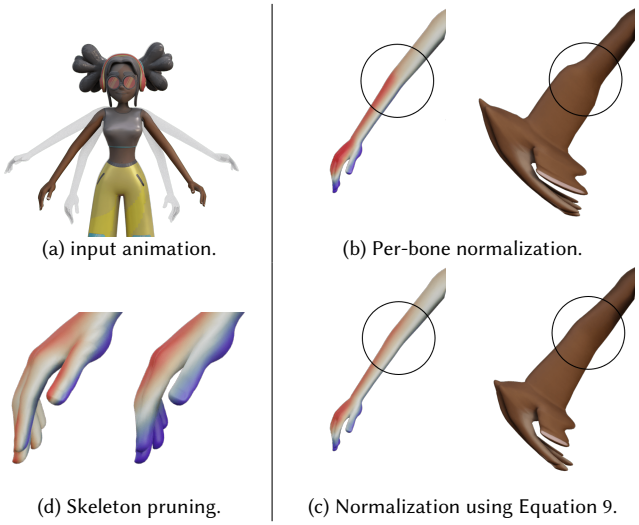
$$\bar{\delta}_{ik} = w_{\text{coll.}}(u_{ik})\frac{\delta_{ik}}{M_{ik}} \quad \text{with} \quad \delta_{ik} = (\mathbf{p}_i - \mathbf{c}_{k,r})\cdot\hat{\mathbf{n}}_k(u_{ik}), \tag{8}$$

where u_{ik} is the parametric position of vertex i along bone k of root $c_{k,r}$ and axis $\hat{\mathbf{b}}_k$, and $\mathbf{n}_k$ is obtained by applying Equation 5 to bone k. The purpose of the collinear weight term $w_{\text{coll.}}$ is to avoid shearing artifacts that occur when the local motion direction is collinear with the bone axis, as illustrated in Figure 7 with a simple mesh animated via a pair of bones.

The motion offset normalization term M_{ik} is given by:

$$M_{ik} = u_{ik}\max_{j\in\mathcal{V}_{k,r}}|\delta_{jk}| + (1-u_{ik})\max_{j\in\mathcal{V}_{k,t}}|\delta_{jk}|, \tag{9}$$

where $\mathcal{V}_{k,r}$ (resp. $\mathcal{V}_{k,t}$) is the set of vertices affected by bones that are connected to the root (resp. the tip) of bone k, including bone k itself. The rationale behind Equation 9 is to ensure that M_{ik} varies smoothly across bone joints. As shown in Figure 8b, using a per-bone normalization (i.e., $\max_{j\in\mathcal{V}_k}|\delta_{jk}|$ with $\mathcal{V}_k$ the set of vertices affected by bone k) produces more abrupt transitions of motion



(a) input animation. (b) Per-bone normalization. (d) Skeleton pruning. (c) Normalization using Equation 9.

Fig. 8. Motion offsets computed for the animation in (a) with different normalization factors (b-c). In (d), we compare the motion offsets computed considering all bones (left) with those obtained by considering the hand as a single body part affected by the forearm bone (right).

offsets around articulations, which show up in the final stylized motion. Note that this smoothing step can create motion offsets that slightly go beyond the $[-1, 1]$ interval at joints between two adjacent body parts with very different shapes. However, this is not an issue in practice since normalization is mainly used to ensure that motion stylization remains predictable over the whole mesh.

Using Equation 7, the level of detail of motion offsets is tied to the complexity of the skeleton. In some cases, one might want to have simpler motion offsets. A good example is the creation of elongated in-betweens for an arm motion, where the artist might want to deform the hand as a whole instead of deforming it at the finger level. We grant such an artistic control by optionally toggling off bones in the skeleton for the computation of motion offsets. All its vertices are then instead affected by its parent bone. This is illustrated in Figure 8d on the character's hand.

4 STYLIZED MOTION

We now rely on motion offsets to control three types of motion stylizations. Our main focus is on the creation of elongated in-betweens (Section 4.1), but we also show how to adapt our method to produce multiple in-betweens (Section 4.2) and motion lines (Section 4.3) in 3D. In accordance with guideline **G3**, the stylization functions introduced in Equations 10, 11 and 13 are not fixed but open to customization, as demonstrated in Section 5.

4.1 Elongated In-betweens

Elongated in-betweens are produced by displacing every mesh vertex according to its motion offset along its trajectory. Vertex trajectories are modeled as a concatenation of Catmull-Rom splines [Catmull and Rom 1974], one per pair of contiguous frames in the input animation. Formally, we write $Q_{i,f}(t)$ the trajectory of vertex i between frames f and $f+1$, with $t \in [0, 1]$ the spline parameter. Tangents between contiguous splines are set to ensure C^1 continuity.

Moving a vertex along its trajectory by an arbitrary displacement $\beta \in \mathbb{R}$ yields a new point $\mathbf{p}'_i(f) = Q_{i,f(\beta)}(t(\beta))$, where $f(\beta) = f + \lfloor\beta\rfloor$ and $t(\beta) = \beta - \lfloor\beta\rfloor$, with $\lfloor x\rfloor$ the floor of x. The floor function is used to deal with values of β outside $[0, 1]$. For instance, at frame $f = 5$ with $\beta = 1.4$, interpolation starts at frame $f + \lfloor\beta\rfloor = 6$ and uses an interpolation parameter $t(\beta) = 0.4$. To convey motion, we apply a displacement $\beta_i(f)$ at vertex i and frame f, controlled by a general stylization function of motion offsets that may take an arbitrary number of arguments "args":

$$\beta_i(f) = \mathcal{S}_E(\bar{\delta}_i(f); \text{args}). \tag{10}$$

Since the displacements β_i are not constrained to the $[-1, 1]$ range, the deformed object may be elongated farther than the previous or next frame in the animation. A simple example of stylization function is $\mathcal{S}_E(\delta; \beta_{\max}) = \delta\beta_{\max}$, with $\beta_{\max} \in \mathbb{R}^+$ an artist-controlled maximum elongation parameter. The elongated in-between presented in Figure 3c is created using $\mathcal{S}_E(\delta; 1)$. More complex styles are demonstrated in Section 5, based on additional parameters such as speed, shape, or surface location.

4.2 Multiples In-betweens

In our approach, we create multiple in-betweens by *adding* displaced copies of the whole mesh in the past and/or the future, optionally followed by opacity modulations of each copy. Formally, a multiple is identified by its index $m \in [-m_p, m_n]$ $(m \neq 0)$, with m_p (resp. m_n) the number of copies in the past (resp. in the future). Each multiple m is displaced using the trajectory-based mechanism of Section 4.1, except that we use the same $\beta_m(f) = m\Delta f$ for all mesh vertices, with Δf an artist-controlled temporal spacing parameter.

The opacity $\alpha_{i,m}(f)$ of a vertex i for a multiple $m \neq 0$ at frame f is then controlled by a general stylization function of motion offsets:

$$\alpha_{i,m}(f) = \mathcal{S}_M(\bar{\delta}_i(f), m; \text{args}). \tag{11}$$

A simple example is obtained by creating the same number of copies in the past and future $(m_s = m_p = m_{\max})$, spaced half a frame apart $(\Delta f = \frac{1}{2})$, and using the following stylization function:

$$\mathcal{S}_M(\delta, m; m_{\max}) = H(\delta m) S\left(\frac{|\delta| - \delta_m}{1 - \delta_m}\right) \text{ with } \delta_m = \frac{|m| - 1}{m_{max}}, \tag{12}$$

where H is the Heaviside function and S the smoothstep function. The first term ensures that vertices with positive (resp. negative) motion offsets are transparent for multiples in the past (resp. in the future). The second term assigns an opacity gradient based on motion offsets, which begins at increasingly higher offset values δ_m the farther a multiple is from the mesh at the current frame. Multiple in-betweens created with $\Delta f = \frac{1}{2}$ and $\mathcal{S}_M(\delta, m; 2)$ are presented in Figure 3d. As explored in Section 5, varying the number of multiples, their spacing, or their opacity through different stylization functions achieves different motion effects.

4.3 Motion lines

Motion lines are created in our approach by tracing trajectory curves using thin tubular meshes in 3D. A subset of surface points is first selected in pre-process to act as seeds for motion lines. In practice,

we use a random selection of mesh vertices for simplicity, but any other solution could be employed. At a given frame f, a motion line ℓ thus starts on the animated mesh at the seed point $\mathbf{p}_\ell(f)$, and is traced along its trajectory until it reaches the end point $\mathbf{p}'_\ell(f) = Q_{\ell,f(\tau)}(t(\tau))$, with $\tau \in \mathbb{R}^+$ a length parameter.

The motion line length $\tau_\ell(f)$ for a curve ℓ at frame f is controlled by a general stylization function of motion offsets:

$$\tau_\ell(f) = \mathcal{S}_L(\bar{\delta}_\ell(f); \text{args}). \tag{13}$$

Note that in this case motion offsets $\bar{\delta}_\ell$ are not computed from vertex positions $\mathbf{p}_i$ but from seed positions $\mathbf{p}_\ell$ in Equation 8.

A simple example of stylization function is $\mathcal{S}_L(\delta; \tau_{\max}) = \delta\tau_{\max}$, with $\tau_{\max} \in \mathbb{R}^+$ an artist-controlled maximum length parameter. Motion lines traced with $\mathcal{S}_L(\delta; 1)$ are shown in Figure 3e. In this case, the other motion line parameters (thickness, color and opacity) are kept constant; but they may also be made to vary with motion offsets or other object properties through more complex stylization functions, as shown in Section 5.

5 RESULTS

Our implementation in Blender is described in Section 5.1 and provided in supplemental material, along with the scene files used to produce all the results of Section 5.2 but Figure 18.

5.1 Implementation

Our Blender implementation consists of two components.

First, motion offsets as described in Section 3 are computed through a Python add-on using the Blender API. They are stored as vertex attributes and must be recomputed efficiently every time the animation is modified. We also precompute the Catmull-Rom splines $Q_{i,f}$ during this step. As reported in Table 1, the whole process is on the order of a few seconds (e.g., roughly 3 seconds for the animation in Figure 11a, with 40 frames and 14267 vertices) and is thus compatible with iterative refinements of the animation (**G2**). It could be further accelerated with a lower-level (e.g., C++) multi-threaded implementation, but the distribution of our Blender add-on would then be less practical. Note that our discussions with professional artists confirmed that a few seconds of pre-processing is common for such tools and is thus not a strong limitation for artists. Table 1 also reports memory usage per frame for storing the Catmull-Rom splines and the motion offsets.

Once the motion offsets are computed, the stylization method presented in Section 4 is implemented with Blender "Geometry Nodes". We provide preset node graphs implementing a given stylization function and exposing simple controls (e.g., sliders) for novice artists and fast prototyping, with interactive feedback provided in the Blender viewport (**G2**). More experimented artists can freely create their own node graphs or customize existing ones to define new styles (**G3**). We provide several examples in the next section. Note that all stylization parameters may be keyframed, as done in some examples shown in the supplemental video.

Table 1. Pre-processing time and memory usage per frame on a computer with an Intel Core i9-7920X 2.90GHz CPU.

	Simple objects		**Articulated characters**	
# vertices	482	1344	1344	14267
# bones	–	–	2	65
Time / frame	**4** ms	**5** ms	**7** ms	**70** ms
Memory usage / frame	**10** KB	**27** KB	**27** KB	**280** KB

5.2 Application Examples

All the results presented in this section are available as animations in the accompanying video, as they are better appreciated in motion. The "Spot" cow model (Figure 9) is courtesy of Crane et al. [2013], articulated characters and animations were downloaded from Mixamo [Adobe 2024]. The basket ball used in Figures 3 and 10 is downloaded from [Downdate 2012]. The peanut model from Figure 4 is downloaded from [Credomo 2016]. Tex Avery's Daffy Duck 3D model is downloaded from [SteveTheDragon 2019]. The flower and tree models in Figure 17 are from [Rohmer et al. 2021]. Spintop model from Figure 19 is downloaded from [JuanG3D 2017]. The cut-out animation in Figure 18 is courtesy of Joel Graham - "Sketchy Squirrel".

Simple object. We present in Figure 9b the result produced by our method when creating elongated in-betweens for a simple object without armature, using $\beta_i = \mathcal{S}_E(\bar{\delta}_i; 1)$ where $\mathcal{S}_E$ is the simple stylization function from Section 4.1. Our method stretches the object to cover adjacent frames, hence easing retinal persistence (**G1**) and smoothing motion. This example of elongated smearing is obtained in one click, which helps novice animators give life to otherwise rigid animations.

Figure 10 shows an example of a smear effect using multiple in-betweens, illustrating the impact of a proper choice for the spacing parameter Δ_f. Indeed, in this case, retinal persistence is achieved when at least one copy of the object appears at the same position between two consecutive frames. To ensure that constraint, we restrict the spacing parameter to $\Delta f(N_\cap) = \frac{1}{m_p+m_n+1-N_\cap}$, where $N_\cap \in [1, m_p + m_n]$ is the desired number of overlapping multiples among consecutive frames. Figure 10 shows a simple example: in the bottom row, we create overlapping multiples with $m_p = m_n = 1$ and $N_\cap = 2$, yielding $\Delta f = 1$; whereas in the top row, we directly set $\Delta f = 0.7$. The latter configuration leads to a stroboscopic effect as shown in the supplemental video. In this example, we use the default stylization from Section 4.2 to compute opacity: $\alpha_{i,m} = S_M(\bar{\delta}_i, m; 1)$.

Articulated character. Figure 11 illustrates the generation of elongated in-betweens on an articulated character for different choices of stylization function. The default function $\mathcal{S}_E(\bar{\delta}_i; 2)$ is shown as a reference in Figure 11b. As seen in the supplemental video, it produces a "wobbly" motion effect, as if the character was made of jelly. To counterbalance this effect, we weight vertex displacement by velocity, which accentuates (resp. dampens) elongation for fast (resp. slow) moving vertices. This is shown in Figure 11c, where we use $\mathcal{S}_E^{\text{speed}}(\bar{\delta}_i; \mathbf{v}_i, \beta_{\max}) = \beta_{\max}\|\mathbf{v}_i\|\bar{\delta}_i$. We further stylize the elongated in-between by locally modulating displacement by a noise value. We use a 3D Voronoi noise [Worley 1996] applied to the position $\mathbf{p}_i(0)$ of vertex i at the start frame of the animation: $F_1(\mathbf{p}_i(0), \kappa)$, where κ is a controllable scaling factor. The stylization function becomes: $\mathcal{S}_E^{\text{noise}}(\bar{\delta}_i; \mathbf{v}_i, \kappa, \beta_{\max}) = \beta_{\max}\|\mathbf{v}_i\|F_1(\mathbf{p}_i(0), \kappa)\bar{\delta}_{i-}$, where

$\bar{\delta}_{i-} = \min(\bar{\delta}_i, 0)$ is used to restrict the noisy smearing effect to trail behind the character, as shown in Figure 11d.

Up until now we have only modulated the motion offset magnitudes. As done by Schmid et al. [2010], one may also want to shift motion offsets toward the past or the future. To this end, we use the warping function $C_{s,p} : [0,1] \to [0,1]$ introduced by Hise [2020], with $s \in [0,1]$ and $p \in [0,1)$ controlling the strength and direction of warping respectively: $p < 0.5$ (resp. $p > 0.5$) warps in the future (resp. the past). The warped motion offsets are then given by $\bar{\delta}'_i(s,p) = 2\,C_{s,p}\left(\frac{\bar{\delta}_i+1}{2}\right) - 1 \in [-1,1]$. They may be used in any stylization function, for example to create elongated in-betweens in Figure 12.

Examples of multiple in-betweens and motion lines on an articulated character are shown in Figure 1. Multiple in-betweens are positioned with the overlap constraint for retinal persistence, using $N_\cap = 1$, $m_p = 2$ and $m_n = 1$, which yields $\Delta f = \frac{1}{3}$. In this case, opacity is controlled by a modified version of the default stylization function (Equation 12) applied to warped motion offsets: $\alpha_{i,m} = H(\|\mathbf{v}_i\| - v_{\min})\mathcal{S}_M(\bar{\delta}'_i(0.8, 0.5), m; 1)$, where the Heaviside term H assigns a 0 opacity to vertices moving slower than $v_{\min}$. Similarly, we use a modified version of the default stylization function for motion lines to compute their length: $\tau_\ell = H(\|\mathbf{v}_\ell\| - v_{\min})\mathcal{S}_L(\bar{\delta}_\ell; 10)$.

Advanced controls. As shown in Figure 13, applying elongated smearing to all the bones of a skeleton might result in an excessive amount of deformation on thin limbs, such as fingers of the hand. Skeleton pruning deals with this potential issue by assigning vertices of the hand to their parent bone. As a result, the whole hand is elongated instead of individual fingers (see also Figures 1 and 12). Pushing this idea to its extreme, one may want to compute the motion offsets of an articulated character using the method described in Section 3.1. This is particularly useful when the whole body is projected in space, as shown in Figure 14.

Stylizing the motion of thin and long objects such as the sword in Figure 15 is best done by adding a single dedicated bone that runs along its length, using $\bar{\delta}_{i-}$ since stylization toward the past is common for sword animations. We complement these motion offsets with manually painted weights $\omega_i \in [0,1]$ that increase along the length of the sword. We show results for all three motion stylizations: elongated in-betweens are obtained with $\beta_i = \omega_i \mathcal{S}_E^{\text{speed}}(\bar{\delta}_{i-}; \mathbf{v}_i, 15)$; multiple in-betweens use $m_p = 4$, $m_n = 0$, $N_\cap = 1$ (yielding $\Delta f = \frac{1}{4}$) and $\alpha_{i,m} = \mathcal{S}_M(10\|\mathbf{v}_i\|\omega_i\bar{\delta}_{i-}, m; 1)$; and motion line lengths are set to $\tau_\ell = \mathcal{S}_L(\|\mathbf{v}_\ell\|\omega_\ell\bar{\delta}_{\ell-}; 20)$, while their opacity decreases linearly from start to end. We additionally displace the seed points $\mathbf{p}_\ell$ along their trajectory to leave a gap between motion lines and the sword.

Stylization is easily adapted to work with a reduced frame rate, which gives the impression of a scene animated "on twos", as commonly done in traditional 2D workflows. This merely requires to scale β_i (for elongated in-betweens), β_m (for multiples in-betweens) or τ_ℓ (for motion lines) by the frame rate step. Examples are shown in the supplemental video for elongated in-betweens.

Additional results. All three motion stylization effects may be used at once, as shown in Figure 16. Our method may also be applied to static scenes observed through a moving camera, by computing motion offsets and vertex trajectories in camera space instead of world space. Figure 17 shows a simple example with elongated in-betweens, using per-object stylization functions: $\beta_i = \mathcal{S}_E(\bar{\delta}_{i-}; 1)$ for the cow and the flower; $\beta_i = \mathcal{S}_E^{\text{noise}}(\bar{\delta}_{i-}; \mathbf{v}_i, 10, 1)$ for the tree.

Our framework works natively on 2D cut-out animations that rely on textured meshes, as shown in Figure 18 with elongated in-betweens on a 2D character.

6 DISCUSSION AND PERSPECTIVES

We have presented a simple yet effective approach to motion stylization in 3D based on motion offsets, and we have demonstrated its adequacy to produce elongated in-betweens, as well as multiple in-betweens and motion lines, all in 3D with interactive feedback.

Relationship to previous work. Stylizing motion directly in 3D has two advantages: (1) output meshes may be manually edited at every frame to fine-tune the style; (2) stylization does not interfere with shading and rendering since they occur later in the pipeline.

Compared to swept volumes (Figure 9c), our method better preserves the local shape of the mesh and is several orders of magnitude faster than the state-of-the-art algorithm of Sellan et al. [2021] (about 7 seconds/frame for swept volumes vs. 0.007 seconds/frame for our motion offsets). In addition, since the topology of the input mesh is preserved, texture naturally follows mesh deformation, whereas swept volumes require recomputing new UV mappings.

The method of Schmid et al. [2010] applies motion stylization (multiple in-betweens and motion lines) at the rendering stage. We believe that their method and ours do not target the same audience: manipulating appearance parameters such as opacity or line thickness in 2D might appeal more to compositing artists; whereas a 3D stylization technique is more adapted to the work of 3D animators as it naturally integrates in their workflow. We thus consider that there is potential interest for these two different approaches.

Limitations & Future work. Our algorithm to compute motion offsets requires the knowledge of the full animation, which makes it impractical for real-time applications such as video games. However, if smear frames are only computed in the past, it could be adapted for such a use case, provided previous vertex positions can be efficiently evaluated or cached.

Moreover, our motion offsets computation relies on a structure that summarizes the object motion, i.e., the centroid for simple shapes and the skeleton for characters. Our method is thus unsuitable for visual effects such as fluids, cloth or hair, usually animated by physical simulations, unless their main motion can be abstracted by such a structure.

Since our elongated in-betweens are obtained by deformation of the mesh geometry, their smoothness depends on both the input tessellation and the amplitude of the displacements. In practice, we used global subdivision passes of mesh to produce smoother results. Yet, to reduce geometric complexity, it would be relevant to develop an automatic adaptive tessellation scheme based on the gradient of the displacements.

Depending on the strength of the rotation component in the input motion, our method may produce elongated in-betweens with self-intersections, that may (or may not) produce visual artifacts in the final render images. The most pathological cases are surfaces

of revolution rotating around their axis of symmetry. In such a case, a simple ad hoc workaround consists in removing the rotation from the object motion for smear stylization and reintroducing it in UV-space to animate a texture, as illustrated in Figure 19. A more general solution would require automatically decomposing the input motion into a set of transformations and letting the artist decide what component to stylize, which is an interesting avenue for future work.

Finally, while motion lines and cartoon stylization have been shown to help observers understand and predict animated motion [Burr and Ross 2002; Garcia et al. 2008; Geisler 1999], no similar user experiment has been conducted for elongated in-betweens. Besides, as already noted in Section 5.2, we observed in our results that uniform smearing may produce a "wobbly" motion effect which modifies the perception of the object's mechanical properties. An interesting future direction would thus be to explore the impact of smear frames on motion and material perception.

ACKNOWLEDGEMENTS

We are very grateful to Quentin Auger from "Dada ! Animation" for his help during and after the interview. We are also grateful to Amélie Fondevilla, Samuel Bernou, Romain Vergne, Damien Rohmer, Melvin Even and Panagiotis Tsiapkolis for helpful discussions. This work is supported by the ANR MoStyle (ANR-20-CE33-0002).

REFERENCES

Adobe. 2024. Mixamo. https://www.mixamo.com/ Accessed: 16-01-2024.

Otman Benchekroun, Jiayi Eris Zhang, Siddartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. 2023. Fast Complementary Dynamics via Skinning Eigenmodes. *ACM Trans. Graph.* 42, 4, Article 106 (2023). https://doi.org/10.1145/3592404

David C Burr and John Ross. 2002. Direct evidence that "speedlines" influence motion mechanisms. *Journal of Neuroscience* 22, 19 (2002), 8661–8664. https://doi.org/10.1523/JNEUROSCI.22-19-08661.2002

Edwin Catmull and Raphael Rom. 1974. A class of local interpolating splines. In *Computer aided geometric design*. Elsevier, 317–326. https://doi.org/10.1016/B978-0-12-079050-0.50020-5

Stephen Chenney, Mark Pingel, Rob Iverson, and Marcin Szymanski. 2002. Simulating cartoon style animation. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering*. 133–138. https://doi.org/10.1145/508530.508553

Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable objects alive! *ACM Trans. Graph.* 31, 4 (2012), 1–9. https://doi.org/10.1145/2185520.2185565

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10. https://doi.org/10.1145/2461912.2461986

Credomo. 2016. https://blendswap.com/blend/17808

James E Cutting. 2002. Representing motion in a static image: constraints and parallels in art, science, and popular culture. *Perception* 31, 10 (2002), 1165–1193.

Downdate. 2012. https://opengameart.org/content/basket-ball-texture

Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sýkora. 2017. Example-Based Expressive Animation of 2D Rigid Bodies. *ACM Trans. Graph.* 36, 4, Article 127 (2017). https://doi.org/10.1145/3072959.3073611

Marcos Garcia, John Dingliana, and Carol O'Sullivan. 2007. A Physically Based Deformation Model for Interactive Cartoon Animation.. In *VRIPHYS*. 27–34. https://doi.org/10.2312/PE/vriphys/vriphys07/027-034

Marcos Garcia, John Dingliana, and Carol O'Sullivan. 2008. Perceptual evaluation of cartoon physics: accuracy, attention, appeal. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization*. 107–114. https://doi.org/10.1145/1394281.1394301

Wilson S Geisler. 1999. Motion streaks provide a spatial code for motion direction. *Nature* 400, 6739 (1999), 65–69. https://doi.org/10.1038/21886

Jason Hise. 2020. https://www.desmos.com/calculator/3zhzwbfrxd

Naoya Iwamoto, Hubert PH Shum, Longzhi Yang, and Shigeo Morishima. 2015. Multi-layer Lattice Model for Real-Time Dynamic Character Deformation. In *Computer Graphics Forum*, Vol. 34. Wiley, 99–109. https://doi.org/10.1111/cgf.12749

N. Jones and J. Keyser. 2005. Real-time geometric motion blur for a deforming polygonal mesh. In *International 2005 Computer Graphics*. 26–31. https://doi.org/10.1109/CGI.2005.1500362

JuanG3D. 2017. https://sketchfab.com/3d-models/day31-spinning-top-d090504b2d994d7c812c14bc963afe90

Niranjan Kalyanasundaram, Damien Rohmer, and Victor Zordan. 2022. Acceleration Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters. In *Graphics Interface*.

Rubaiat Habib Kazi, Tovi Grossman, Cory Mogk, Ryan Schmidt, and George Fitzmaurice. 2016. ChronoFab: Fabricating Motion. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 908–918. https://doi.org/10.1145/2858036.2858138

Ji-yong Kwon and In-Kwon Lee. 2012. The Squash-and-Stretch Stylization for Character Motions. *IEEE Transactions on Visualization and Computer Graphics* 18, 3 (2012), 488–500. https://doi.org/10.1109/TVCG.2011.48

John Lasseter. 1987. Principles of traditional animation applied to 3D computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 35–44. https://doi.org/10.1145/37402.37407

Jiaju Ma, Li-Yi Wei, and Rubaiat Habib Kazi. 2022. A Layered Authoring Tool for Stylized 3D animations. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. Article 383, 14 pages. https://doi.org/10.1145/3491102.3501894

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph.* 24, 3 (2005), 471–478. https://doi.org/10.1145/1073204.1073216

Paul Noble and Wen Tang. 2007. Automatic expressive deformations for implying and stylizing motion. *The Visual Computer* 23 (2007), 523–533. https://doi.org/0.1007/s00371-007-0125-8

Richard Roberts and Byron Mallett. 2013. A pose space for squash and stretch deformation. In *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*. IEEE, 166–171. https://doi.org/10.1109/IVCNZ.2013.6727010

Damien Rohmer, Marco Tarini, Niranjan Kalyanasundaram, Faezeh Moshfeghifar, Marie-Paule Cani, and Victor Zordan. 2021. Velocity Skinning for Real-time Stylized Skeletal Animation. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.142654

Nadine Abu Rumman and Marco Fratarcangeli. 2014. Position based skinning of skeleton-driven deformable characters. In *Proceedings of the 30th Spring Conference on Computer Graphics*. 83–90. https://doi.org/10.1145/2643188.2643194

Johannes Schmid, Robert W Sumner, Huw Bowles, and Markus H Gross. 2010. Programmable motion effects. *ACM Trans. Graph.* 29, 4 (2010), 57–1.

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2021. Swept volumes via spacetime numerical continuation. *ACM Trans. Graph.* 40, 4 (2021), 1–11. https://doi.org/10.1145/3450626.3459780

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. 245–254. https://doi.org/10.1145/325334.325242

SteveTheDragon. 2019. https://sketchfab.com/3d-models/daffy-duck-57b3d5631e4649da907977353aece0c8

Frank Thomas and Ollie Johnston. 1981. Disney animation: The illusion of life. (1981).

Jue Wang, Steven M Drucker, Maneesh Agrawala, and Michael F Cohen. 2006. The cartoon animation filter. *ACM Trans. Graph.* 25, 3 (2006), 1169–1173. https://doi.org/10.1145/1141911.1142010

Richard Williams. 2001. *The Animator's Survival Kit*. Faber & Faber.

Steven Worley. 1996. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 291–294. https://doi.org/10.1145/237170.237267

Jiayi Eris Zhang, Seungbae Bang, David I.W. Levin, and Alec Jacobson. 2020. Complementary Dynamics. *ACM Transactions on Graphics* (2020). https://doi.org/10.1145/3414685.3417819

Xiuming Zhang, Tali Dekel, Tianfan Xue, Andrew Owens, Qiurui He, Jiajun Wu, Stefanie Mueller, and William T Freeman. 2018. Mosculp: Interactive visualization of shape and time. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 275–285. https://doi.org/10.1145/3242587.3242592

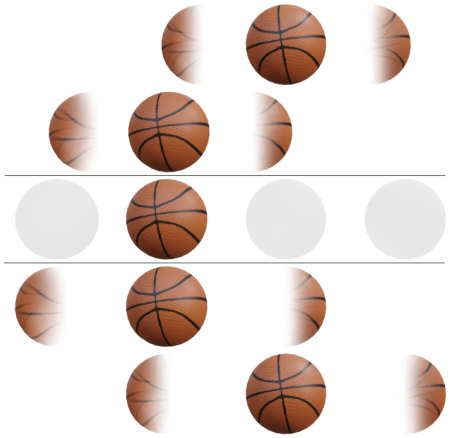
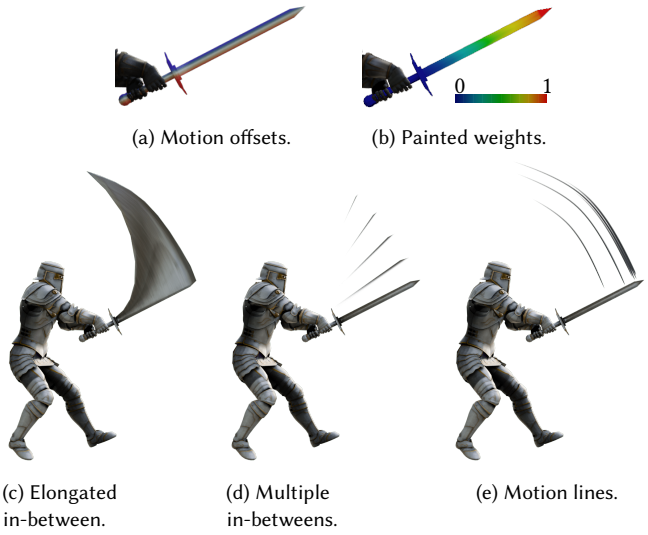


Fig. 10. Multiple in-betweens created for a basket ball translating from left to right (middle), with overlapping multiples for retinal persistence (bottom), and without this constraint (top).

Fig. 12. Left: elongated in-between created with $\mathcal{S}_E^{\text{speed}}(\bar{\delta}_i; \mathbf{v}_i, 3)$. Right: same stylization with warped motion offsets $\bar{\delta}_i'(0.5, 0.8)$ to create a time shift effect towards the past. Color-coded motion offsets before and after warping are shown in insets.

Fig. 13. Left: Elongated in-betweens generated with motion offsets computed considering the hands as single body parts instead of each finger independently, and with the stylization function $\mathcal{S}_E^{\text{speed}}(\bar{\delta}_i, \mathbf{v}_i, 6)$. Right: Close-up on the right hand, (a) stylized with these motion offsets and (b) with the motion offsets of Figure 11c.



(a) Motion offsets. (b) Painted weights.

(c) Elongated in-between. (d) Multiple in-betweens. (e) Motion lines.

Fig. 15. Various stylization effects (c-e) of a sword slash motion using (a) motion offsets computed with a single bone to control the axis of separation, and (b) manually painted weights to control stylization intensity.

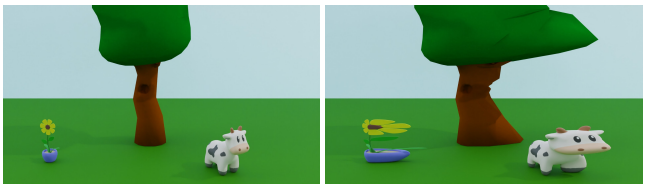


Fig. 17. Left: static scene observed by a camera translating from left to right. Right: when stylizing objects using elongated in-betweens by considering their motion with respect to the camera, the speed of the observer is conveyed through deformations, as if the scene was seen from a train window.

Fig. 18. Left: 2D cut-out animation (courtesy of Joel Graham - "Sketchy Squirrel"). Right: elongated in-between created on the cat's tail using $\beta_i = \mathcal{S}_E(\bar{\delta}_i; 3)$.

(a) Input animation. (b) Elongated in-betweens. (c) Swept volumes.

Fig. 9. Using (a) the animation of a simple object, we compare (b) elongated in-betweens generated by our method with the simple stylization function $\beta_i = \bar{\delta}_i$, to (c) swept volumes of the object along its trajectory between previous and next frames, computed using the method of Sellan et al. [2021].

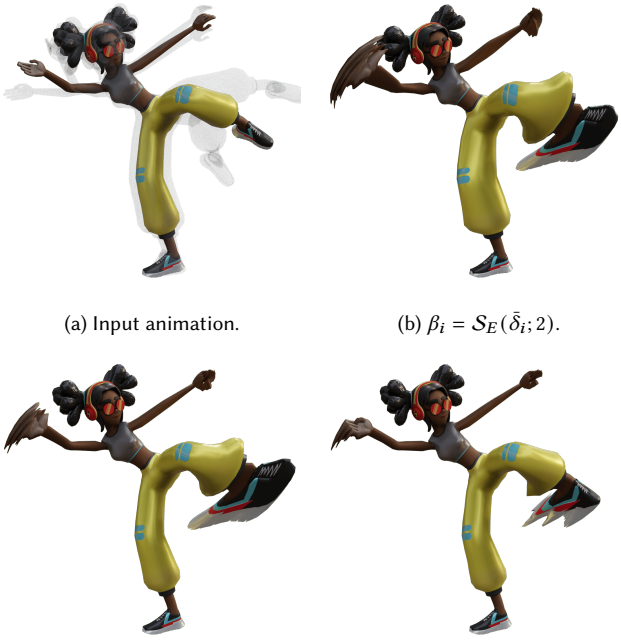
(a) Input animation with onion skin visualization of adjacent frames.

(b) With skeleton-based motion offsets

(c) With motion weights computed on the full body motion, with method from Section 3.1

Fig. 14. Motion offsets and elongated in-between for a frame of a full body motion, with $\beta_i = \mathcal{S}_E(\bar{\delta}_i; 0.7)$. Using weights computed on the whole body as a single object instead of at the skeleton level, we obtain a different stylization that emphasizes the global motion of the character.

Fig. 16. The stylization effects achievable by our method may be combined to create complex motion stylizations.



(a) Input animation. (b) $\beta_i = \mathcal{S}_E(\bar{\delta}_i; 2)$.

(c) $\beta_i = \mathcal{S}_E^{\text{speed}}(\bar{\delta}_i; \mathbf{v}_i, 6)$. (d) $\beta_i = \mathcal{S}_E^{\text{noise}}(\bar{\delta}_{i-}; \mathbf{v}_i, 18, 10)$.

Fig. 11. Elongated in-betweens created with different stylization functions for the middle frame of a character animation.

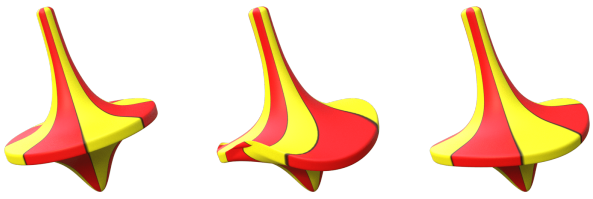


Fig. 19. Left: the motion of symmetrical objects may raise issues in our approach. Middle: the rotation component around the axis of revolution creates artifacts (self-intersections) when the motion is stylized by elongated in-betweens. Right: a work-around consists in replacing the rotation around the axis by a translation in UV-space (effectively rotating the texture around the axis), which yields a more coherent motion stylization.