Interactive Art-Directable Smear Frame Stylization

Jean Basset, Pierre Bénard, Pascal Barla

Journées Françaises de l'Informatique Graphique 2023 / J. Martínez Bayona and N. Faraj.
(Paper Chairs)

Interactive Art-Directable Smear Frame Stylization

Jean Basset, Pierre Bénard and Pascal Barla
Inria, Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, France

Figure 1: We present a method that stretches an animated object, including articulated characters, along its space-time trajectory to create smear frames, with interactive stylistic control for art direction.

Abstract

Smear frames are used by artists to expressively convey motion in an animation. In this paper, we present a method to generate elongated inbetweens, i.e., smear frames created by stretching an object along its space-time trajectory. Our method takes as input the mesh of an animated character, with or without a skeleton, and outputs a deformed version of the mesh for each frame of the animation, that can still be refined by an artist before rendering. We first compute spatially and temporally coherent displacement weights that correspond to the direction and intensity of stretching of each vertex of the mesh. We then describe a framework to stylize these displacement weights, customizing elongated inbetweens at interactive rates. We propose built-in stylization functions that can be used by novice users for simple and fast smear frame generation, but our Blender "Geometry node" implementation allows easy integration of custom-made stylization functions designed by more expert artists.

1. Introduction

2D animators have long explored techniques to convey motion expressively, such as motion lines inspired from comic books and smear frames to simulate motion blur. Smear frames encompass two main effects: *multiples*, that correspond to stroboscopic copies of a drawing, and *elongated inbetweens*, which consist in stretching the animated subject to cover the span of several frames in fast motions [Ric01]. The latter is widely used in traditional 2D animation to put emphasis on motion trajectories (see Figure 2). Back in 1987, Lasseter [Las87] described a similar effect using manual *squash-and-stretch* deformations. Garcia et al. [GDO08] showed that such deformations effectively make motion more predictable for observers, and numerous works explored creation and customization of these effects [CPIS02, KL11, RTK*21, MWK22]. Smear frames, and other motion effects, recently regained popularity with 3D animated movies with strong stylistic identities such as the "Spider-verse" movies.

In this work we are specifically interested in recreating elongated smear frames in 3D animations. Schmid et al. [SSBG10] proposed programmable motion effects, that allow the creation of smears in different styles (e.g., speed lines, multiples, directional blur). However, these effects emerge in screen-space during the rendering process, and can only be edited through motion programs. We aim to directly deform the geometry of the animated object to create elongated inbetweens that can still be adjusted by the artist before

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

rendering, without the need for a custom renderer. Closer to our approach, swept volumes have been used to depict 3D motion in static images and sculptures [KGM*16, ZDX*18], and can be used to create elongated inbetweens [JK05, SAJ21]. However, since a swept volume is the union of all points spanned by an object during an animation, it is not guaranteed to be homeomorphic to the input object: some concavities and high frequency details of the original surface may be hidden, potentially impeding recognition of the object. Moreover, expensive computation times make art direction difficult. In contrast, we aim to deform the input geometry while preserving its topology and surface details, and to provide interactive feedback to artists.

In this paper, we present a framework to generate elongated inbetweens for keyframe-based animation of 3D meshes with stylistic control (see Figure 1). Our core contribution is a method to compute spatially and temporally coherent displacement weights on the mesh corresponding to surface points that should be displaced towards the past or the future along their space-time trajectories. The amount of displacement can then be freely stylized and customized by an artist at interactive rates to produce the final stretched object. Self-intersections aside, the resulting mesh has the same topology as the input by construction. The supplemental video and Blender files to experiment with our system will be available on the project homepage https://mostyle.github.io/.

2. Method

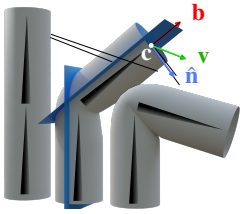
Our method takes as input an animated polygonal mesh whose topology is fixed throughout the animation. For articulated characters, skeleton and skinning weights are additionally provided. As a pre-process, we first compute for all frames a displacement weight at each vertex of the mesh, corresponding to the intensity and direction (past or future) of stretching of this vertex along its trajectory (Section 2.1). We then propose a framework that deforms the surface mesh based on these displacement weights, and allows the artist to freely control them to create various smear styles (Section 2.2).

2.1. Displacement Weights

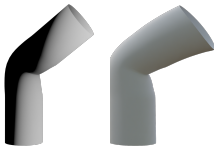
For every frame f, we compute for each vertex with index i a displacement weight $\delta_i(f) \in [-1,1]$. During the creation of the elongated inbetweens, vertices will be displaced along their trajectory by a signed distance controlled by $\delta_i(f)$. When $\delta_i(f) = 1$ (resp. -1), the vertex i is displaced to its position in the next (reps. previous) frame of the animation; this normalization makes art-direction more predictable. We observed in traditional 2D animations that parts of an object that are "leading" (resp. "tailing") the motion are sent farther in the future (resp. the past), while parts close to the center of the object have small displacements. Similarly to Jones et al. [JK05], a naive approach to reproduce this behavior would be to measure how much a given vertex is facing the motion, i.e., considering the angle between its normal and velocity. However, this metric is not spatially coherent for objects with concavities, in particular meshes with high-frequency details have rapidly varying normal directions in a neighborhood, which results in noisy δ values.

To overcome this limitation, we partition the object into a forward and a backward region using a secant plane (see Figure 3). For articulated characters, we process each body part independently, but to obtain deformations that best preserve the shape of the object, we additionally enforce the secant plane to pass through the bone axis. This ensures spatial continuity of the displacement weights of adjacent body parts (see Figure 3c). A downside of this constraint is that the orientation of the plane does not accurately represents motion collinear with the axis of the bone. This can create artifacts such as shearing, as illustrated in Figure 4b. To alleviate this problem, we reduce displacement weights according to motion collinearity (Figure 4c).

Single bone objects. We define the instantaneous direction of motion $\hat{\mathbf{v}}$ as the normalized mean velocity of the mesh vertices. We



(a) Secant planes (in blue) for a simple two-bones capsule bending while translating left to right.

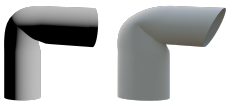


(b) Without bone axis constraint.

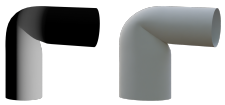
(c) With bone axis constraint.

Figure 3: Secant planes (a) computed by our method; resulting displacement weights and deformations without (b) and with (c) constraining the plane to pass through the bone axis. Positive (resp. negative) weights are painted white (resp. black).

(a) Two bones capsule translating left to right.



(b) Without collinear weights.



(c) With collinear weights.

Figure 4: Displacement weights computed for the animation in (a) when weighting δ_i according to motion collinearity (c), and without this weighting scheme (b). Displacement weights are depicted on the left (with the same color scheme as in Figure 3) and the corresponding deformation on the right.

then orient the secant plane in such a way that it is as orthogonal as possible to the mean velocity while respecting the above constraint (see Figure 3), that is the plane passing through one of the bone extremities and whose normal $\hat{\mathbf{n}}$ is defined by:

$$\mathbf{n} = \hat{\mathbf{v}} - (\hat{\mathbf{v}} \cdot \mathbf{b})\mathbf{b}, \quad \hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}, \tag{1}$$

with $\mathbf{b}$ the normalized bone direction.

The farther a vertex is from this secant plane, the more it will be displaced in the past or the future depending on which side of the plane it sits. Denoting $\mathbf{p}_i$ the position of the i^{th} vertex, its displacement weight δ_i is thus its signed distance to the secant plane:

$$\delta_i = (\mathbf{p}_i - \mathbf{c}) \cdot \hat{\mathbf{n}}, \tag{2}$$

with $\mathbf{c}$ the position of a point on the bone axis. We then normalize the displacement weights of each body part such that the highest absolute displacement is 1, and reweigh them according to motion collinearity:

$$\hat{\delta}_i = \omega_{\text{collinear}} \frac{\delta_i}{\max_i(|\delta_i|)} \quad \text{with} \quad \omega_{\text{collinear}} = 1 - (\mathbf{b} \cdot \hat{\mathbf{v}})^2. \tag{3}$$

For simple objects without skeleton, we can relax the bone constraints, in which case $\hat{\mathbf{n}} = \hat{\mathbf{v}}$, $\mathbf{c}$ is the centroid of the object and $\omega_{\text{collinear}} = 1$.

Articulated characters. For articulated characters, simply using these displacement weights computed per body part might still result in abrupt changes in intensity or even direction of elongation around bone joints. Similarly to [RTK*21], we use the skinning weights to enforce spatial continuity. The final $\bar{\delta}_i$ of a vertex i is the weighted sum of its displacement weights computed for each influencing bone, using the skinning weights. As the skinning weights of a given vertex must sum to 1, these displacement weights still respect $\bar{\delta}_i \in [-1, 1]$.

For animations with fast changing motion directions, displacement weights of certain vertices may vary abruptly in time, creating stroboscopic artifacts. To alleviate this issue, we temporally smooth the displacement weights using a temporal window of k frames around the current frame f, that is:

$$\bar{\delta}_i(f) = \sum_{j \in \{-k, \cdots, k\}} \left(1 - \left(\frac{j}{k+1}\right)^2\right)^2 \hat{\delta}_i(f+j). \tag{4}$$

We use $k = 2$ in all our experiments.

2.2. Elongated Inbetweens Creation

The displacement weights are then used as input parameters of a stylization function $\mathcal{S}$ that computes the final intensity of displacement $\beta_i(f)$ of each vertex along its space-time trajectory to create the elongated inbetweens:

$$\beta_i(f) = \mathcal{S}(\bar{\delta}_i(f), \text{args}). \tag{5}$$

The stylization function $\mathcal{S}$ may take an arbitrary number of arguments depending on the stylization, such as geometric or contextual information, and its outputs are not constrained to the $[-1, 1]$ range.

To obtain smooth deformations, we estimate the space-time trajectory of every vertex using Catmull-Rom splines [CR74], which allows smooth interpolation of vertex positions at any time $t \in [0, 1]$ between two consecutive frames. For each vertex i at frame f, we compute its stretched position

$$\mathbf{p}'_i(f) = Q(i, f, \beta_i(f)), \tag{6}$$

where $Q(i, f, t)$ is the Catmull-Rom interpolation of the position of vertex i at frame f and $f+1$ by a factor $t \in [0, 1]$. Catmull-Rom spline join at the extremities, forming a C^1 continuous curve. For β_i outside the $[0, 1]$ range, we shift the starting frame of the interpolation (e.g.,for $f = 1$ and $\beta_i = 1.4$, we compute $\mathbf{p}'_i$ as $Q(i, 2, 0.4)$).

Our method outputs a deformed mesh at each frame of the animation, with the same connectivity as the input mesh. The result is still editable by the artist before rendering.

3. Results

Our prototype is implemented in Blender as a Python plugin for the displacement weights computation (Section 2.1), and as "Geometry nodes" for the stylization (Section 2.2). The latter grants interactive and intuitive manipulation of per-vertex parameters such as our displacement weights.

We present in Figure 5b results when directly applying $\bar{\delta}_i(f)$ to the animation (i.e., $\beta_i(f) = \bar{\delta}_i(f)$). Similarly, in Figure 6, we compare results with $\beta_i = \bar{\delta}_i$ for a simple object without armature, and to swept volumes computed by the method of Sellán et al. [SAJ21]. To obtain elongated inbetweens with this approach, we replace each frame of the animation by the swept volume of the current frame and one adjacent frame in the past and future. As discussed earlier, the swept volume hides concavities and details of the object, making it harder to recognize. Moreover, as our method does not change the connectivity of the mesh, the texture naturally follows the deformation of the inbetweens, while the swept volumes would require adapting the texture to the new geometry.

In the remainder of this section, we propose examples of stylization functions $\mathcal{S}$ to showcase potential use cases of our approach. See supplemental video for better visualizations. Articulated characters are extracted from Adobe's Mixamo [mix].

Speed weighting. Our method naturally communicates the speed of different parts of an animated character. An interesting stylization would be to exaggerate this effect. However simply multiplying displacement weights by a scalar would equally increase the stretching of fast and slow moving vertices, which can result in distorted perception of materials (e.g., a "wobbly" effect, best seen in a video). Instead, we can insert back the speed of each vertex as a weighting factor in the stylization function, which will exaggerate stretching of fast moving vertices while dampening it for slow vertices:

$$\mathcal{S}_{\text{speed}}(\bar{\delta}_i, \mathbf{v}_i, \alpha_{\text{speed}}) = \alpha_{\text{speed}} \|\mathbf{v}_i\| \bar{\delta}_i, \tag{7}$$

with $\mathbf{v}_i$ the velocity of vertex i, and α_{speed} a user-control parameter. We present results of this stylization in Figure 5c.

Noise. A classical stylization of smear frames consists in stretching the animated object according to a noise texture. To generate a similar effect, we evaluate a 3D Voronoi noise [Wor96]

(a) Input animation with onion skin visualization of adjacent frames. (b) Applying directly the displacement weights $\beta_i = \bar{\delta}_i$. (c) Applying the stylization function $\beta_i = \mathcal{S}_{\text{speed}}(\bar{\delta}_i, \mathbf{v}_i, 6)$. (d) Applying the stylization function $\beta_i = \mathcal{S}_{\text{noise}}(\bar{\delta}_i^-, \mathbf{p}_i(f_0), 18, 3)$.

Figure 5: Elongated inbetweens created with different stylization functions for the middle frame of a character animation.

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

(b) Result by our method, with $\beta_i = \bar{\delta}_i$. (c) Result using swept volumes [SAJ21].

Figure 6: Elongated inbetweens created for an animation of a simple object without skeleton (a), using our method (b) and a swept volume approach (c).

$F_1(\mathbf{p}_i(f_0), s)$ on the vertex positions $\mathbf{p}_i(f_0)$ at the first frame of the animation f_0 to ensure temporal coherence. The factor s controls the size of the underlying implicit grid. We then multiply the displacement weights by the noise value:

$$\mathcal{S}_{\text{noise}}(\bar{\delta}_i, \mathbf{p}_i(f_0), s, w) = w \,.\, F_1(\mathbf{p}_i(f_0), s)\bar{\delta}_i \tag{8}$$

We present a result with this stylization in Figure 5d. In this figure, we also used $\bar{\delta}_i^- = \min(\bar{\delta}_i, 0)$ to limit stretching to the past, creating noisy smears trailing the object.

Painted weights. Our method allows the user to finely customize smear frames by painting per-vertex scalar weights ω_i and using the stylization function $\mathcal{S}_{\text{painted}}(\bar{\delta}_i, \omega_i) = \bar{\delta}_i \,.\, \omega_i$. We present a result of this stylization in Figure 7.

Stepped animation. Another common strategy used by animators to mimic the traditional 2D feel is to use stepped animation [ÓC23, For23], e.g., animating "on twos" by dropping every other frame. As a lower framerate implies that consecutive frames

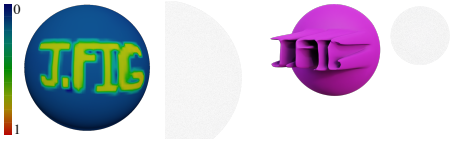


Figure 7: Weights manually painted by the user (left), and corresponding weighted elongated in-between generated with the stylization function $\mathcal{S}_{\text{painted}}$ for a simple 3D translation (right).

are farther apart, stroboscopic artifacts can appear for fast motions. Smear frames are then a natural solution to preserve the smoothness of the animation [Las87]. Our method allows to customize stepped animations, by choosing a framerate (e.g., on twos, on threes), dropping frames accordingly and displaying kept frames for several consecutive frames. Stylization parameters β_i are scaled accordingly so that the kept frames are stretched to cover the dropped frames. Illustrations of our method applied to stepped animation are presented in the supplementary video.

4. Discussion

In this paper, we presented an approach to create elongated inbetweens by stretching an animated object along its space-time trajectory. After a pre-process taking a few seconds, our method allows for interactive artistic control to create a wide range of styles.

Further exploration would be needed to validate our framework, e.g., discussions with artists or usability studies. More specifically, user studies would be important to validate the visual quality of our results according to observers, and to explore the impact of potential artifacts such as self-intersections. Moreover, while motion lines and cartoon stylization have been shown to help observers understand motion [Gei99, BR02, GDO08], to the best of our knowledge, no similar studies have been done on elongated inbetweens. An interesting future direction would thus be to explore the impact of our stylization on motion perception, and to compare how observers react to different styles of smear frames such as elongated inbetweens, multiples, swept volumes, or motion lines.

References

[BR02] BURR D. C., ROSS J.: Direct evidence that "speedlines" influence motion mechanisms. *Journal of Neuroscience 22*, 19 (2002), 8661–8664. 4

[CPIS02] CHENNEY S., PINGEL M., IVERSON R., SZYMANSKI M.: Simulating cartoon style animation. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering* (2002), pp. 133–138. 1

[CR74] CATMULL E., ROM R.: A class of local interpolating splines. In *Computer aided geometric design*. Elsevier, 1974, pp. 317–326. 3

[For23] FORREST B. J.: The time filter system: Advanced time manipulation for animators and effects artists: How i learned to stop worrying and love matrix math. In *ACM SIGGRAPH 2023 Talks* (2023), ACM. 4

[GDO08] GARCIA M., DINGLIANA J., O'SULLIVAN C.: Perceptual evaluation of cartoon physics: accuracy, attention, appeal. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization* (2008), pp. 107–114. 1, 4

[Gei99] GEISLER W. S.: Motion streaks provide a spatial code for motion direction. *Nature 400*, 6739 (1999), 65–69. 4

[JK05] JONES N., KEYSER J.: Real-time geometric motion blur for a deforming polygonal mesh. In *International 2005 Computer Graphics* (2005), pp. 26–31. 2

[KGM*16] KAZI R. H., GROSSMAN T., MOGK C., SCHMIDT R., FITZMAURICE G.: Chronofab: fabricating motion. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), pp. 908–918. 2

[KL11] KWON J.-Y., LEE I.-K.: The squash-and-stretch stylization for character motions. *IEEE transactions on visualization and computer graphics 18*, 3 (2011), 488–500. 1

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 35–44. 1, 4

[mix] Adobe's mixamo. https://www.mixamo.com/. Accessed: 12-10-2023. 3

[MWK22] MA J., WEI L.-Y., KAZI R. H.: A layered authoring tool for stylized 3d animations. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022), pp. 1–14. 1

[ÓC23] ÓSKARSSON M. Ö., CHAZOT M.: Spider-man: Across the spider-verse-how to build a pipeline for variable rate animation. In *ACM SIGGRAPH 2023 Talks*. 2023, pp. 1–2. 4

[Ric01] RICHARD W.: *The Animator's Survival Kit*. Faber & Faber, 2001. 1

[RTK*21] ROHMER D., TARINI M., KALYANASUNDARAM N., MOSHFEGHIFAR F., CANI M.-P., ZORDAN V.: Velocity skinning for real-time stylized skeletal animation. *Computer Graphics Forum* (2021). 1, 3

[SAJ21] SELLÁN S., AIGERMAN N., JACOBSON A.: Swept volumes via spacetime numerical continuation. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–11. 2, 3, 4

[SSBG10] SCHMID J., SUMNER R. W., BOWLES H., GROSS M. H.: Programmable motion effects. *ACM Trans. Graph. 29*, 4 (2010), 57–1. 1

[Wor96] WORLEY S.: A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 291–294. 3

[ZDX*18] ZHANG X., DEKEL T., XUE T., OWENS A., HE Q., WU J., MUELLER S., FREEMAN W. T.: Mosculp: Interactive visualization of shape and time. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), pp. 275–285. 2