Automatic Intermediate Frames for Stroke-based Animation

Nicolas Barroso, Amélie Fondevilla, David Vanderhaeghe

Journées Françaises de l'Informatique Graphique 2021 / R.Vergne and N.Mellado.
(Paper Chairs)

Automatic Intermediate Frames for Stroke-based Animation

Nicolas Barroso†, Amélie Fondevilla, David Vanderhaeghe

IRIT, Université de Toulouse, CNRS, UT3, INP, Toulouse, France.

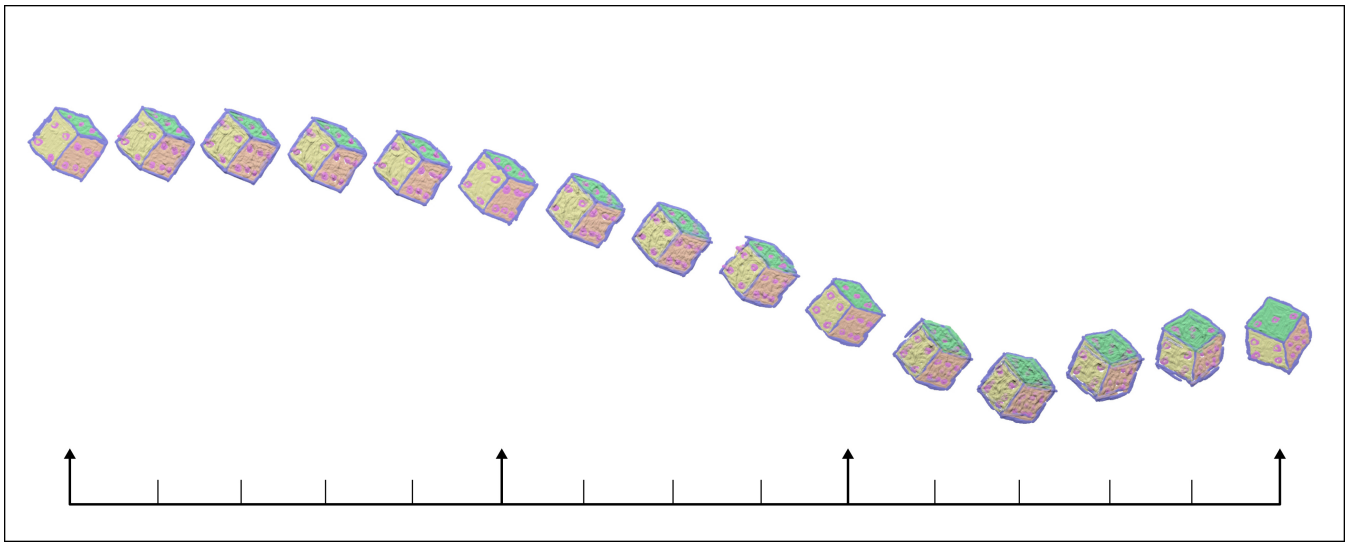

Figure 1: *From a user provided set of key frames (designated by arrows) and motion field, our approach generates intermediate to create an animation.*

Abstract

Creating a 2D animation with visible strokes is a tedious and time consuming task for an artist. Computer aided animation usually focus on cartoon stylized rendering, or is built from an automatic process as 3D animations stylization, loosing the painterly look and feel of hand made animation. We propose to simplify the creation of stroke-based animations: from a set of key frames, our methods automatically generates intermediate frames with coherent motion. Each intermediate frame looks as it could have been drawn by an artist, using the same high level stroke based representation as key frame, and in succession they display the subtle temporal incoherence usually found in hand-made animations.

CCS Concepts

•***Computing methodologies*** → ***Non-photorealistic rendering;*** *Animation;*

1. Introduction

2D animated feature film exhibits a wide range of visual appearances. In this paper we focus on appearance that shows individual 2D marks as in sketching or painting. In this kind of style we distinguish two main workflows. The classical workflow is to create key frames to define the movement, and then create the necessary inbetweening to smooth this movement. The other workflow, known as "under the camera", is to create a first frame, then slightly modify this frame to obtain the next one, and so on. Each frame of the animation only exists when its captured and is then destroyed to produce the next one. The under the camera workflow needs specific artistic skills to anticipate the whole animation sequence. Computers algorithm can help artists in many tedious animation tasks, improving the efficiency of the pipeline in various ways [GM19]. Our

† E-mail: nicolas.barroso@irit.fr

long term goal is to provide computer tools to have the look and feel of the under the camera workflow with the level of control of the classical workflow.

This paper focus on the automatic creation of intermediate frames given a set of key frames. We target professional animators and need to provide fine control over each frame, even those computed automatically. To this end we use stroke based rendering [VC12] where each stroke of the animation is represented independently, rather than keeping only the pixels representation of the frames. Strokes are rendered using a paint simulator to compute the final frame appearance.

The main contributions of the paper are:

- the extrapolation of 3D rendered motion field to obtain a 2D motion field suitable for stroke propagation (Sec 4),
- an optimization scheme to generate guide strokes from key frames and motion field (Sec 5),
- a frame generator that leverage guide stroke from up to two key frames to produce intermediate frames (Sec. 6).

These contributions allows us to elaborate a first inbetween pipeline to generate intermediates frames in a stroke based rendering framework.

2. Related Works

Synthesis of Intermediate frames. The automatic synthesis of intermediate frames, also called inbetweening, is a well-known domain in the field of 2D animation. Most methods work in image space, by synthesizing texture depicting as-rigid-as-possible deformation [SDC09, BCK*13, DBB*17], or using a triangulation of the image space and applying deformation directly on this triangulated mesh, using 2D skinning techniques [BKLP16]. Texture synthesis approaches are agnostic to the tools used to produce the example. The resulting frames are raster images. Our approach generates strokes that give high level structure for post editing. Closer to our method, other approaches are working at the stroke level, deducing the deformation between two frames through stroke pairing [WNS*10], or using directly the motion of an underlying 3D animation [WDK*12]. While most of the stroke-based approaches are focused on sketches, a few of them deal with paintings. In this case, the inbetweening problem can be overcome by embedding the painting strokes in a 3D space [SSGS11], where the strokes can follow the motion of a 3D surface [BBS*13]. We propose to define all the strokes in the 2D image which corresponds to the traditional animation pipeline.

Temporal coherence. Temporal coherence plays a significant role in the perception of stylized animations. In texture approach it can be ensure by energy minimization [BCK*13] or neural network architectures [RDB18]. To get closer to the hand drawn look some works try to reduce the temporal coherence of the stylization process and inject randomness through noise control in their algorithm [FLJ*14] [FJS*17] or use marks approach to obtain a more finer control of the randomness [CZBB20]. As chown by Delanoy et. al. [DBH19], processing the motion rather than the image, with motion rigidifaction for instance, break temporal coherence provided by the initial optical flow and results in 2D look closer to manual methods. Our approach focus on automatic, yet plausible, intermediate frame generation, with the noise inherent to hand made animation, rather than enforcing temporal coherence of the animation.

Generation of strokes by example. Chen et. al. [CZBB20] builds strokes of intermediate frames by rigid transformation of the strokes of key frames. Intermediate frames are composed by selecting and transforming strokes to reach a target density map. The density map is estimated by interpolating key frame's densities with the image registration method [SDC09]. Combined with transformed strokes the animation sequence shows an hand drawn look. We share the same approach, but in our case case, the strokes' transformation are guided by an underlying motion field that gives the expected motion. In addition, our approach modifies the curve shape using optimization scheme that preserve geometric properties from the key frames.

3. Overview

Key frames refer to frame drawn, by the artist, and *generated frames* are automatically rendered by our approach. The animation workflow is as follow: the artist draw one or more key frames which convey the example style and appearance to reproduce (Fig. 1). The artist also provides a motion field that encompass the animation motion. Our approach generates intermediate frames taking key frames and motion, one frame after the other (Fig. 2). At any time, the artist can decide that frame do not corresponds to his wishes and modify or redraw it, this introduce a new key frame in the animation. Intermediate frames are updated to take into account this new key frame. For the rest of the paper we consider that intermediate frames are surrounded by two key frames, one before and one after along the timeline.

We focus on the animation of one object, multiple objects and background are handled using layers. The final animation is composed of multiple layers, each layer is animated and rendered independently, using a different motion field and key frames. Layers composition can then be mixed as painted on different transparent canvas, or mixed as if they are paint one over the other.

Key frames and intermediate frames are defined by an ordered list of strokes. A stroke is defined by a curve represented as a polyline, and a set of parameters, i.e. the quantity and color of paint on the virtual tool and the pressure of the tool on the canvas along the

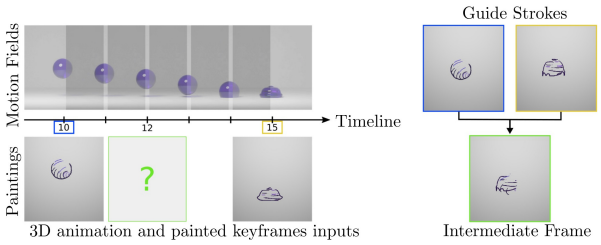

Figure 2: *We generate an intermediate frame, here at time 12, using the guide strokes computed from the key frames before (at time 10) and after (at time 15) on the timeline.*

curve. The final image is obtained by rendering the strokes with a paint simulator. There is two main steps to generate an intermediate frame: the propagation of guide strokes (Sec. 5) using motion field (Sec. 4) and the generation of frame's strokes (Sec. 6).

4. Motion Field

Our method needs a motion field for each frame of the animation as input. This motion field can come from different sources, for instance it can be handcrafted by the artist. In practice, for the example shown in the paper, we start from a simple 3D animation capturing the motion to convey. Since the motion field is created from the render of a 3D scene, we propose to display the 3D scene as a background template to draw a key frame. The proposed UI allows the artist to choose whether to draw or not over the underlying image.

For each frame, the motion field contains two motion vectors for each pixel of the image, one vector to next frame position of this pixel and one vector to previous frame position. Since the motion field is null outside of the animated object's surface, we need to extend the motion information for each background pixels to let the artist paint over the background as well. To this end we compute bi-harmonic weights as describe by Baster et. al. [BBA09] The main step of the algorithm is to build a triangular 2D mesh over the image plane to finally interpolate vertices motion vectors for each pixels. Vertices of this mesh are evenly distributed over the image plane. Each vertex over a surface is assigned the underlying motion vector and become a seed point to the interpolation. The motion vector of vertices falling on background pixels are computed using bi-harmonic weight computed for each vertices according to the seeds, as shown Figure 3. Finally, the motion vector of each pixels of the image plane is computed using barycentric interpolation from the motion field of the triangular mesh.

Moreover we store a confidence value with each motion vector. The confidence is computed by the same barycentric interpolation considering a value of one for each vertices of the 2D mesh that has valid motion and zero for null vectors.

5. Guide Strokes

To guide the generation of an intermediate frame we generate a set of guide strokes. We compute a guide stroke for each strokes of a key frame. A guide stroke captures both key frame content, i.e. the curve, color, and pressure variation, and the motion information, i.e. the motion fields from stroke time to intermediate frame time.

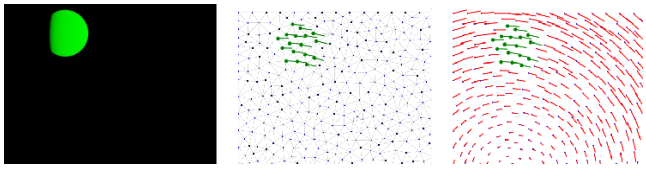

Figure 3: *Computed motion field of a bouncing ball. Left: motion field as computed from the 3D renderer. Middle: 2D mesh and known vectors in green. Right: computed motion vector (in red) from the known vectors using bi-harmonic interpolation.*

Let's consider one stroke of one key frame at time t_0, as stated before, the stroke's curve is described by a poly line, we call the curve of the stroke in the key frame the *reference curve*. The advected curve is computed by moving each poly-line vertex according to underlying motion field. Advection only is not sufficient, it can break the curve shape and also spread curve away from the underlying object due to the motion field interpolation. We regularize the stroke curve using an optimization.

We leverage the confidence value to optimize the guide stroke curve. The optimization process mix the reference curve, with the advected curve according to confidence value: closer to reference curve when the confidence is low ; or closer to the advected curve when confidence is high.

We use the following three energy functions. These energies are measured along the curve for N evenly distributed sample points. The first constraint ensures the optimization will converge spatially close to the advected curve.

$$E_s = \sum_{i=0}^{N-1} \left\| c_{adv}\left(\frac{i}{N}\right) - c_{opt}\left(\frac{i}{N}\right) \right\|^2$$

where $c(p)$ is the evaluation of a 2D point of the parametric curve function at $x \in [0,1]$, c_{adv} is the advected stroke and c_{opt} is the curve of the guide stroke.

The two next energy functions are weighted by the confidence $\sigma(x)$ along the curve of the advected stroke. The second constraint penalizes curvature differences between the reference curve and the curve of the guide stroke:

$$E_c = \sum_{i=0}^{N-1} \left(\gamma_{ref}\left(\frac{i}{N}\right) - \gamma_{opt}\left(\frac{i}{N}\right)\right)^2 \times \left(1 - \sigma\left(\frac{i}{N}\right)\right)$$

where γ is the curvature computed with finite differences.

The last energy ensures that the length between two consecutive samples of the curve of the guide stroke remains close to the length of corresponding samples in the reference curve.

$$\delta(a,b) = \|f(a) - f(b)\|$$

$$E_l = \sum_{i=1}^{N-1} \left(\delta_{ref}\left(\frac{i}{N}, \frac{i-1}{N}\right) - \delta_{opt}\left(\frac{i}{N}, \frac{i-1}{N}\right)\right)^2 \times \left(1 - \sigma\left(\frac{i}{N}\right)\right)$$

Since the stroke poly-line have a lot of points, the optimization would be underconstrained. So, we first fit the poly-line by piecewise cubic Bézier curves, resulting in fewer variables to optimize. We use the Levenberg Marquardt algorithm to solve the optimization. The variables are control points of Bézier curves, and the energy minimized is

$$E = E_s + E_c + E_l$$

The curve regularization is shown Figure 4, where reference curve, advected curve and guide curve are shown for a complex motion.

This process is repeated for each stroke of the key frame. And for each intermediate frame, computing the new advected curve using

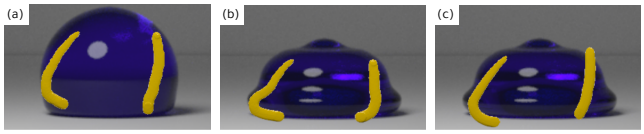

Figure 4: *(a) Example of two reference strokes on a key frame painted in yellow, (b) advected strokes deformed by advection only (c) or modified by our optimization, here with confidence set to 0 everywhere for illustration purpose.*

the previous guide stroke as a starting position. For each intermediate frame we also compute guide strokes from the next key frame, using the same process using backward motion vector.

6. Stroke Generation

The stroke generation stage consists in creating the set of stroke of an intermediate frame given the guide strokes. To do so, we make three straightforward assumptions:

1. The more an intermediate frame is close in time to key frame, the more it should look like this key frame.
2. An intermediate frame should reflect the content of the key frames before and after it.
3. The intermediate frame should have been done by hand, or at least look likes a hand made drawing.

We propose to generate the strokes of an intermediate frame as follow: We select a subset of the guide strokes from the two key frames.

To go from key frame A at time t_a to key frame B at time $t_b = t_a + N$ we progressively select guide strokes generated from key frame B while deselecting stroke generated from key frame A.

To ensure a complete representation of the animated object, we choose to have at least the guide strokes from one of the key frame fully selected at each intermediate frame. To this end we define a selection ratio for intermediate frame at time $t_a + i$ as $\min\left(1, \frac{2(N-i)}{N}\right)$ Selected guide strokes generated from key frame A is 100% for intermediate frames from time t_a to $t_a + N/2$, and decrease to 0% at time t_b. The ratio of guide stroke from key frame B is computed similarly, but reversed in time. We design this selection scheme to ensure correct coverage in the intermediate frames.

We tested two strategies to select the guide stroke given the ratio. The first strategy select a random subset of the guide strokes according to the ratio independently for each intermediate frame. The second strategy randomize the list of guide stroke once for all, then selects the strokes according to the ratio following the list order. The artist can test both strategy and pick the one he prefer.

When rendering the intermediate frame, the stroke drawing order follow the order in the key frames: We assign a scalar value as *draw time* between 0 and 1 to each of the strokes of a key frame, according to the drawing order of the artist. The selected strokes of both key frames are sorted by this draw time before rendering.

7. Implementation and Results

Our C++/OpenGL prototype use Radium Engine [MRB*21] as main rendering engine. We use our own implementation of the paint simulator presented by Baxter et. al. [CBWG10]. This paint simulator compute bi-directional paint exchanges between the brush and the canvas, on the GPU. Strokes advection is done on the GPU using compute shader. The Levemberg Marquardt optimization uses Eigen [GJ*10] implementation. We extract motion fields through Blender AOV rendering [Com18], we use triangle lib in python [She96, R*20] and our implementation of bi-harmonic weight interpolation to obtain the interpolated motion field, as a pre-process. The time needed to render a frame depends on the number of guide strokes generated and the number of pixel covered by the strokes. In practice, the examples shown in the paper and supplemental videos took less than one second per frame to render, including advection, optimization and paint simulation. These render time are suitable for interaction.

8. Conclusion and Discussion

Our method generate intermediate frames in a stroke painterly style and produce an animation with a traditional hand drawn look and feel. It takes as input key frames drawn by an artist, and a motion field to describe how the strokes should move. While we diverge from the traditional paint under the camera look and feel, we ease the creation of animation in a stroke based rendering context by automatically creating intermediate frames.

The main limitation of the approach concern the complexity of the motion we can depict. For instance, when two moving objects cross each others on the same layer, some point of the poly-line of stroke of one object will follow the other object motion. To solve this issue, we think a better registration of each stroke with the underlying motion field could be envisioned.

Several improvements are left as future work, for instance during the generation of the strokes of an intermediate frame, we can imagine to add some randomness in the strokes parameters (curve shape, pressure, brush color). In our current selection scheme, the intermediate frame at mid-time between two key frames render the full set of strokes from the two key frames. The resulting image depict thicker paint on the canvas, which could be perceived as an artifact. Also, the generation of strokes from guides strokes could have a target coverage over the image plane to remove this artifact.

We think that the confidence value could convey artistic expression, so that some strokes keep their original shapes during advection, while others better follow the underlying motion. To this end we plan to propose confidence brushes to alter the confidence of the motion field, either for some regions, or for specific selected strokes.

The creation of an animated 3D scene to compute the motion field adds a large amount of work. We see two diverging workflows here: Either stay in pure 2D animation, and explore new tools to build the motion field from artist gesture, or take more benefit from the 3D animation, such as color, surface normal, and lighting, to make working time profitable.

Acknowledgement This work has been partially supported by the Structures JCJC ANR project ANR-19-CE38-0009-01.

References

[BBA09] BAXTER W., BARLA P., ANJYO K.: N-way morphing for 2d animation. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 79–87. doi:10.1002/cav.310. 3

[BBS*13] BASSETT K., BARAN I., SCHMID J., GROSS M., SUMNER R. W.: Authoring and animating painterly characters. *ACM Trans. Graph. 32*, 5 (Oct. 2013). doi:10.1145/2484238. 2

[BCK*13] BÉNARD P., COLE F., KASS M., MORDATCH I., HEGARTY J., SENN M. S., FLEISCHER K., PESARE D., BREEDEN K.: Stylizing animation by example. *ACM Trans. Graph. 32*, 4 (July 2013). doi:10.1145/2461912.2461929. 2

[BKLP16] BAI Y., KAUFMAN D. M., LIU C. K., POPOVIĆ J.: Artist-directed dynamics for 2d animation. *ACM Trans. Graph. 35*, 4 (July 2016). doi:10.1145/2897824.2925884. 2

[CBWG10] CHU N., BAXTER W., WEI L.-Y., GOVINDARAJU N.: Detail-preserving paint modeling for 3d brushes. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, Association for Computing Machinery, p. 27–34. doi:10.1145/1809939.1809943. 4

[Com18] COMMUNITY B. O.: *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org. 4

[CZBB20] CHEN J., ZHU X., BÉNARD P., BARLA P.: Stroke Synthesis for Inbetweening of Rough Line Animations. In *Pacific Graphics Short Papers, Posters, and Work-in-Progress Papers* (2020), Lee S.-h., Zollmann S., Okabe M., Wuensche B., (Eds.), The Eurographics Association. doi:10.2312/pg.20201233. 2

[DBB*17] DVOROŽŇÁK M., BÉNARD P., BARLA P., WANG O., SÝKORA D.: Example-based expressive animation of 2d rigid bodies. *ACM Trans. Graph. 36*, 4 (July 2017). doi:10.1145/3072959.3073611. 2

[DBH19] DELANOY J., BOUSSEAU A., HERTZMANN A.: Video motion stylization by 2d rigidification. In *Proceedings of the 8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (Goslar, DEU, 2019), Expressive '19, Eurographics Association, p. 11–19. doi:10.2312/exp.20191072. 2

[FJS*17] FIŠER J., JAMRIŠKA O., SIMONS D., SHECHTMAN E., LU J., ASENTE P., LUKÁČ M., SÝKORA D.: Example-based synthesis of stylized facial animations. *ACM Trans. Graph. 36*, 4 (July 2017). doi:10.1145/3072959.3073660. 2

[FLJ*14] FIŠER J., LUKÁČ M., JAMRIŠKA O., ČADÍK M., GINGOLD Y., ASENTE P., SÝKORA D.: Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. *Computer Graphics Forum 33*, 4 (2014), 1–10. doi:10.1111/cgf.12407. 2

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 4

[GM19] GEORGE-MOLLAND A.-L.: Innovation technique dans les studios d'animation et d'effets visuels : la Recherche et Développement au service du pipeline. *La Création Collective au Cinéma 2* (2019), 101–124. URL: https://hal.archives-ouvertes.fr/hal-02133118. 1

[MRB*21] MOURGLIA C., ROUSSELLET V., BARTHE L., MELLADO N., PAULIN M., VANDERHAEGHE D., ET AL.: Radium-engine, July 2021. URL: https://storm-irit.github.io/Radium-Engine/, doi:10.5281/zenodo.5101334. 4

[R*20] RUFAT D., ET AL.: Triangle, 2020. URL: https://rufat.be/triangle/. 4

[RDB18] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision 126*, 11 (2018), 1199–1219. doi:10.1007/s11263-018-1089-z. 2

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2009), NPAR '09, Association for Computing Machinery, p. 25–33. doi:10.1145/1572614.1572619. 2

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Lin M. C., Manocha D., (Eds.), vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry. 4

[SSGS11] SCHMID J., SENN M. S., GROSS M., SUMNER R. W.: Overcoat: An implicit canvas for 3d painting. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, Association for Computing Machinery. doi:10.1145/1964921.1964923. 2

[VC12] VANDERHAEGHE D., COLLOMOSSE J.: Stroke Based Painterly Rendering. In *Image and Video-Based Artistic Stylisation*, Rosin P., Collomosse J., (Eds.), vol. 42 of *Computational Imaging and Vision*. Springer, London, 2012, pp. 3–21. doi:10.1007/978-1-4471-4519-6_1. 2

[WDK*12] WHITED B., DANIELS E., KASCHALK M., OSBORNE P., ODERMATT K.: Computer-assisted animation of line and paint in disney's paperman. In *ACM SIGGRAPH 2012 Talks* (New York, NY, USA, 2012), SIGGRAPH '12, Association for Computing Machinery. doi:10.1145/2343045.2343071. 2

[WNS*10] WHITED B., NORIS G., SIMMONS M., SUMNER R. W., GROSS M., ROSSIGNAC J.: Betweenit: An interactive tool for tight inbetweening. *Computer Graphics Forum 29*, 2 (2010), 605–614. doi:10.1111/j.1467-8659.2009.01630.x. 2