Controllable motion lines generation for an abstracted depiction of 3D motion.

Mohamed-Amine Farhat, Alexandre Bléron, Camille Noûs, Romain Vergne, Joëlle Thollot

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

Controllable motion lines generation for an abstracted depiction of 3D motion.

Amine Farhat[12] Alexandre Bléron[2] Camille Noûs [3] Romain Vergne[1] Joëlle Thollot[1]

[1]MAVERICK
[2]Left-Angle
[3]Laboratoire Cogitamus

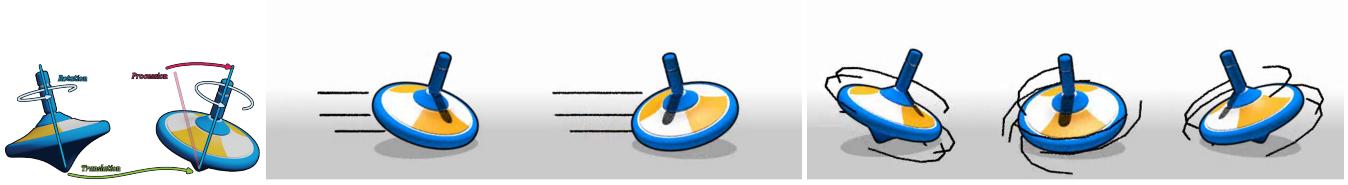

Figure 1: *Based on a user input motion hierarchy (left), our method allows the artist to generate motion lines depicting sub-motions. Here, we show two types of motion lines, generated from the same input animation depicting the translation (middle) or the rotation (right) of a spinning top.*

Abstract

In spite of the increasing popularity of non-photorealistic rendering techniques, very few methods allow to stylize the motion of moving objects. We propose a semi-automatic method to generate motion lines on top of a rendered 3D scene. We start with a decomposition of the motion of the object into simple affine transformations, chosen by the artist. By drawing a proxy curve *on an arbitrary frame, artists control the position of the motion lines in screenspace, relative to the object. We then automatically extend the proxy to the rest of the animation, while respecting the intent of the artist through a set of weighted constraints. Finally, we use this proxy to generate a time-parametrized drawing space, on which the motion lines are generated.*

CCS Concepts

•***Computing methodologies*** → ***Motion processing; Non-photorealistic rendering;***

1. Introduction

Various rendering styles are now available in commercial software and stylization is becoming increasingly popular in the animation industry. However, some typical effects found in traditional animations are still hand-drawn by artists on top of rendered frames. In particular, motion lines design is still a time consuming task whereas they greatly contribute to the perception of the motion in the final animation.

In this paper we propose an automated method to generate expressive motion lines based on a 3D scene. A motion line can be described as a stroke appearing close to a moving object, and representing its motion by abstracting part of its trajectory. By observing traditional motion lines and actual animation movies, we have identified three main properties that we think a motion line creation method must fulfill:

Motion simplification. Artists tend to simplify complex movements for both purposes of readability and expressiveness. This complexity often manifests as movements composed of simpler sub-movements. For instance, as shown in Figure 1, a spinning top's gyration is a composition of a rotation around its axis, a precession, and a planar translation. An artist may choose to emphasize only the translation or only the rotation with motion lines.

Controlling the effect's 2D shape. Motion lines are traditionally shaped in 2D. For instance, when representing a translation, lines keep a constant spacing in 2D. However, ensuring 2D properties becomes problematic when working in a 3D scene. This is a well-known problem in the field of animation stylization: the temporal coherence problem [BBT11]. Thus, a 2D control has to be provided when designing a generation method for motion lines.

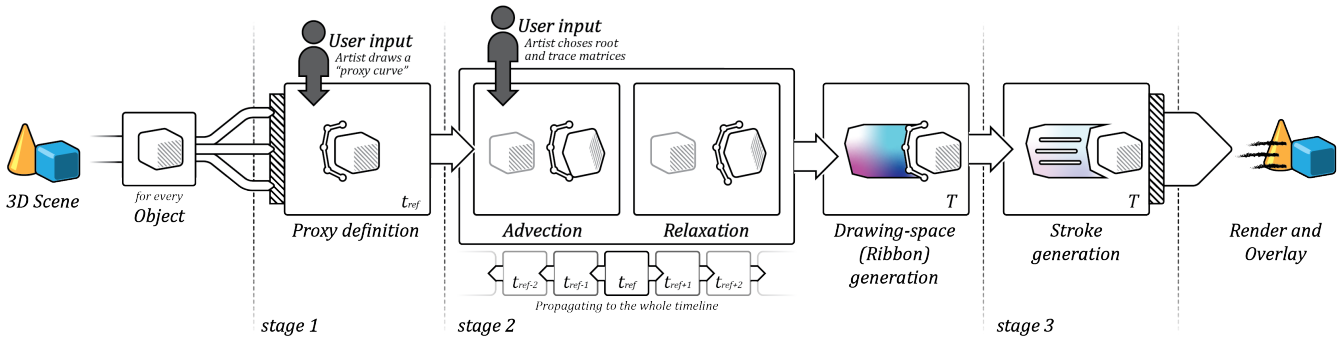

Figure 2: *Overview of our pipeline.*

Style control. There is great variability within line-based motion effects. Artists allow themselves a lot of leeway when it comes to the aspect of their motion lines: they often vary in number, length, closeness to one another or color, depending on the motion's context and the artist's style. Therefore a motion line design method must offer a satisfying level of control over these properties.

In this paper, we propose a controllable motion lines design method. We let the artist design and chose sub-motions based on an input motion decomposition. Our process is based both on 3D and 2D primitives, which allows for an accurate depiction of 3D motion without sacrificing the hand-drawn appearance.

2. Previous works

In [KHK03], Kawagishi et al. were the first to address the rendering of motion lines for stylized animations. Their method targets 2D cel animation. They rely on the trailing edge of the animated 2D object to propose various styles of motion effects that they call cartoon motion blur. Their method works very well for 2D scenes but is hard to adapt to 3D scenes due to the complex behavior of silhouettes in 3D.

Meanwhile, part of Salvati and Ito's method [SI18] allows them to generate an expressive blur effect on moving 2D elements. Their method distinguishes between translations and rotations, and applies the corresponding blur effect. Unfortunately, it is unclear whether they handle translations that do not follow a straight line, since the directional blur they use would be unable to bend with the curved trajectory of an object.

Still working in 2D, Kim et al. [KE05] designed a method to stylize the motion of objects in videos. Their method extracts and tracks the motion of "segments", which are groups of pixels that share similar properties. This model uses 2D primitives, notably "leading" and "trailing" edges. However, these primitives are not parametrized, which greatly limits the artist's control over the position of the motion lines.

Closest to our method, Schmid et al. [SSBG10] are targeting 3D scenes. Their motion abstraction is based on a "time aggregate object" (TAO), allowing them to trace the consecutive positions of any point on a mesh, which they use to generate motion lines. This method produces various effects and can deal with complex scenes; however, it offers no screenspace control when it comes to placing and shaping the lines.

3. Method

3.1. Overview

Our pipeline, illustrated in Figure 2, takes as input a set of 3D objects, their respective transform matrices and their parenting relationships. In the first stage (see Section 3.2), artists choose the motion they want to depict by using the movement decomposition. In a second stage, we compute a parametrized drawing-space. For that, the artist draws a proxy curve for a given frame (see Section 3.3.1). The proxy curve is then automatically extended to the rest of the animation. By evaluating the resulting proxy curves through time we compute a full 3D parametrized surface for each frame, that we call *ribbons* (see Section 3.3.2). The final stage consists in drawing the motion lines taking advantage of the parametrization of the ribbons.

3.2. Motion selection

In order to give the artist complete control over the motion simplification, we propose to rely on the input scene motion decomposition. Such a decomposition is defined by a set of transform matrices. Out of these, artists are free to choose which matrices best express the movement they wish to depict.

When evaluated over the duration of the animation, the product of these chosen matrices generates a *trace* of the trajectory of the object. This product matrix is referred to as the *trace matrix*, denoted M_{trace}.

However, the curves generated by the trace matrix are not guaranteed to be coherent with the object's motion, since they only represent a sub-portion of it (as shown Figure 3). We therefore introduce a second matrix, the *root matrix*, denoted M_{root}, whose purpose is to allow the trace to follow the object.

The product of the trace matrix and the root matrix generates a trajectory that depicts the chosen motion while following the global motion of the object.

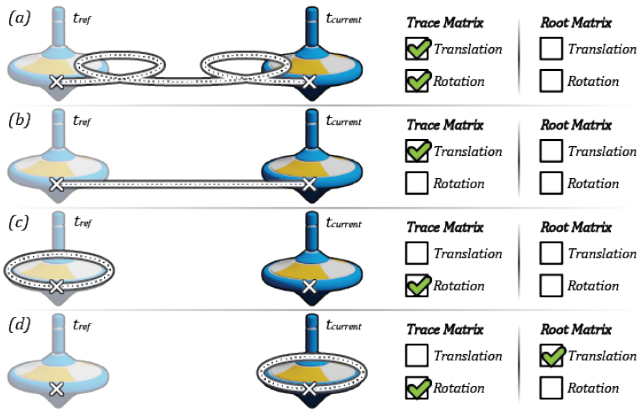

Figure 3: *Examples of different motion selections and the corresponding matrices: (a) Trivial case, the motion is entirely depicted by the trace - we are following a point as it moves in world space. (b) We only represent the translation - this is the motion of the point if the object was not rotating. (c) We only represent the rotation, but the trace does not follow the object as it moves. (d) We set the translation matrix as a root matrix to keep the trace in sync with the object.*

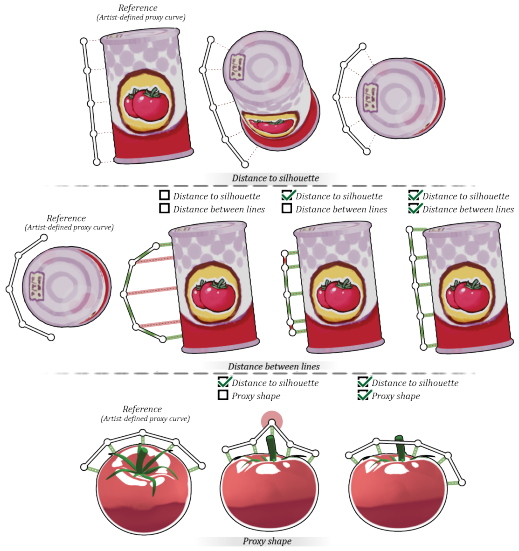

Figure 4: *Three 2D constraints are optimized to propagate the proxy curve along the animation.*

3.3. Drawing-space generation

The purpose of this stage is to create a parametrized 3D surface at each frame, that we call a *ribbon*, on which the motion lines will be generated. The ribbons must (1) allow to delineate the area where the motion lines are drawn, (2) allow a 2D control of the motion lines' shape and positions and (3) accurately abstract the chosen motion. For those purposes, each ribbon is generated by sweeping a user-defined *proxy curve* over time following the trace matrix. The curve serves the delineation and its parametrization is done in screenspace. Using the trace matrix to generate the ribbon ensures a proper depiction of the chosen motion.

3.3.1. Proxy curve generation

The proxy curve is a parametrized 3D curve that is placed by the artist in screen-space at a chosen frame, referred to as the *reference frame*. For example, an artist may draw the proxy curve on the *trailing* side of an item – i.e. on the side of the silhouette facing away from the direction of the motion – in order to depict a translation. In practice, our curve is a polyline with a conservative number of control points. The curve is drawn in 2D by the artist, and is then back-projected onto the scene using a *billboard* plane centered at the object's origin.

Once drawn by the artist, we extend the proxy curve to the rest of the animation, while maintaining coherency with the depicted movement and ensuring the constancy of 2D properties. In this paper, we have chosen to target three common 2D constraints illustrated in Figure 4:

1. *Silhouette proximity.* We want the lines to always start at a fixed distance from the object. Doing so greatly contributes to the 2D look of the effect.
2. *Distance between lines.* Ensuring a regular spacing between the motion lines is commonly found in traditionnal animation. For that we need to keep a constant proxy length, a fixed distance between the points of the polyline and therefore a stable parametrization. It greatly increases temporal coherence by reducing jittering and counteracts the inherent tendency of the silhouette proximity constraint to shrink the curves [BJC*12].
3. *Line shape.* There are still cases where the previous constraint does not maintain temporal coherence. We propose a last constraint whose role is to maintain the shape of the curve as it was given by the user by minimizing the changes to its angles.

While such a set of goals is inherently impossible to concurrently fulfill, we take inspiration from active strokes [BJC*12] to design a set of energies to minimize through a gradient descent, so as to compromise between our goals.

Advection. The first step is to advect the proxy curve so as to follow the 3D motion of the object. We apply the $M_{trace} \times M_{root}$ transform to every point on the proxy curve. This amounts to moving the proxy curve according to the object's motion, regardless of the 2D constraints.

Relaxation. We then propose a relaxation step that will modify the shape of the proxy curve to ensure our chosen 2D constraints.

We first compute the energy of the resulting proxy curve for each constraint C_i as the sum of the difference between the constraint at frame t and the constraint at the reference frame t_{ref}:

$$E_i = \sum_{\mathbf{p} \in P} |C_i(\mathbf{p}(t_{ref})) - C_i(\mathbf{p}(t))|$$

with P the set of control points of the proxy curve.

The final constraint energy is the weighted sum of the energies of

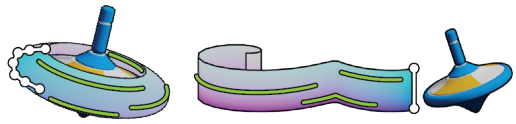

Figure 5: *Here, we show the two different ribbons generated for the spinning top exemple. (Left) A rotation: The proxy takes a cylinder-like shape wrapped around the object. (Right) A translation: The proxy has been drawn on the side of the object facing away from the motion's direction. The ribbon deforms to fit the object's silhouette.*

every constraint. Then, using a gradient descent, we find the closest local minimum of the constraint energy.

This approach does not guarantee a proxy curve that verifies all the constraints perfectly. Instead, we end up on a compromise between all the constraints.

In practice, for each point $\mathbf{p}$ of the proxy polyline we compute the constraint $C_i(\mathbf{p})$ as follows:

1. *Silhouette proximity.* $C_1(\mathbf{p})$ is defined as the distance between $\mathbf{p}$ and its closest point on the silhouette.
2. *Distance between lines.* $C_2(\mathbf{p})$ is defined as the distance between $\mathbf{p}$ and the next point on the polyline.
3. *Line shape.* $C_3(\mathbf{p})$ is defined as the angle formed with its two neighboring points.

3.3.2. Ribbons

Now that we have computed a proxy curve for each frame, we compute our final drawing spaces, the ribbons, by sweeping each proxy curve. For a given frame $t \in T$ we use the root matrix $M_{root}(t)$ to anchor the ribbon to the object and we use the trace matrix $M_{trace}(x)$ to compute the trace of the proxy curve along the full animation, that is for all $x \in T$ and for $y_{\mathbf{p}}$ the parameter of the point $\mathbf{p}$ on the proxy polyline :

$$R_t(x, y_{\mathbf{p}}) = \mathbf{p}\, M_{trace}(x)\; M_{root}(t) \quad \forall\, x \in T, \mathbf{p} \in P$$

$R_t(x,y)$ is then a curved surface representing the drawing space at a given time t. The two coordinates x and y can be respectively seen as a temporal dimension and a spatial dimension: y being the arc length along the initial proxy curve, evaluated at a frame x. Since the transform operations applied to each proxy curve are affine, the surface maintains a coherent parametrization.

By choosing the trace matrix, the artist controls the shape of the ribbon and thus the shape of the final motion lines as shown Figure 5. Drawing a simple straight line in ribbon-space already yields satisfying results. Indeed, the shape of the line is determined by the curvature of the ribbon's surface which, by construction, always represents the targeted motion.

4. Implementation

We have implemented our method as a JavaScript web-application. This choice was motivated by our willingness to make our prototype readily available for testing purposes. The matrix decomposition is provided as a JSON file. We use a blender python script to export the matrices of every object in a scene and their respective hierarchy. All of the parameters described in this paper are available and editable through the UI. The proxy can be drawn directly on the viewport. The rendering operation can usually be completed in interactive time; however, since our implementation is CPU-based, performances could be greatly improved.

5. Results

Figure 1 shows a first example of motion lines generated by our method. In the following, we show results obtained by varying the line drawing parameters and taking advantage of the ribbons' parametrization.

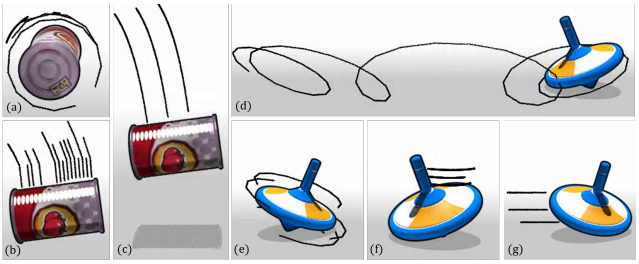

Figure 6: *Several examples of motion lines: An object falling and rolling with rotation lines with random lengths (a), or a chosen number of translation lines (b and c). A spinning top with a depiction of its global motion (d), or only its rotation (e), precession (f) or translation (g).*

6. Discussion and conclusion

We have demonstrated that our method is suitable for procedurally generating expressive motion lines without sacrificing the artist's agency over their placement and shape.

Our method takes advantage of the ribbon's parametrization to give control to the artists. However, our ribbon generation can still suffer from temporal incoherency. We plan to address this issue by adding a regularization term to our energy minimization that should take into account the neighbouring frames when computing the proxy curve and the ribbon's final shape. We would also like to explore *temporal* decomposition. For instance, if an object sharply changes direction during a translation, it could be useful to have two different sets of motion lines for the two directions, mainly for expressivity purposes.

Expecting the artist to draw a proxy curve can sometimes be unmanageable, and exceedingly so in scenes with a flurry of moving objects. Having an alternative *procedural* way of generating the initial proxy curve would greatly contribute to the applicability of our method to more use-cases.

Finally, in the future, we plan to add a style module to our method that would allow the artist to render the generated motion lines with a large variety of styles.

References

[BBT11] BÉNARD P., BOUSSEAU A., THOLLOT J.: State-of-the-art report on temporal coherence for stylized animations. *Computer Graphics Forum 30*, 8 (2011), 2367–2386. URL: https://theses.hal.science/tel-00630112, doi:https://doi.org/10.1111/j.1467-8659.2011.02075.x. 1

[BJC*12] BÉNARD P., JINGWAN L., COLE F., FINKELSTEIN A., THOLLOT J.: Active strokes: Coherent line stylization for animated 3d models. In *NPAR 2012-10th International Symposium on Non-photorealistic Animation and Rendering* (2012), ACM, pp. 37–46. 3

[KE05] KIM B., ESSA I.: Video-based nonphotorealistic and expressive illustration of motion. pp. 32– 35. doi:10.1109/CGI.2005.1500363.

[KHK03] KAWAGISHI Y., HATSUYAMA K., KONDO K.: Cartoon blur: nonphotorealistic motion blur. pp. 276 – 281. doi:10.1109/CGI.2003.1214482. 2

[SI18] SALVATI M., ITO K.: Blur algorithms for cartoon animation. pp. 1–2. doi:10.1145/3283289.3283295. 2

[SSBG10] SCHMID J., SUMNER R., BOWLES H., GROSS M.: Programmable motion effects. *ACM Trans. Graph. 29* (07 2010). doi:10.1145/1833349.1778794.