CurveCrafter: A System for Animated Curve Manipulation

Nora S Willett
Pixar Animation Studios

Kurt Fleischer
Pixar Animation Studios

Haldean Brown
Pixar Animation Studios

Ilene L E
Pixar Animation Studios

Mark Meyer
Pixar Animation Studios

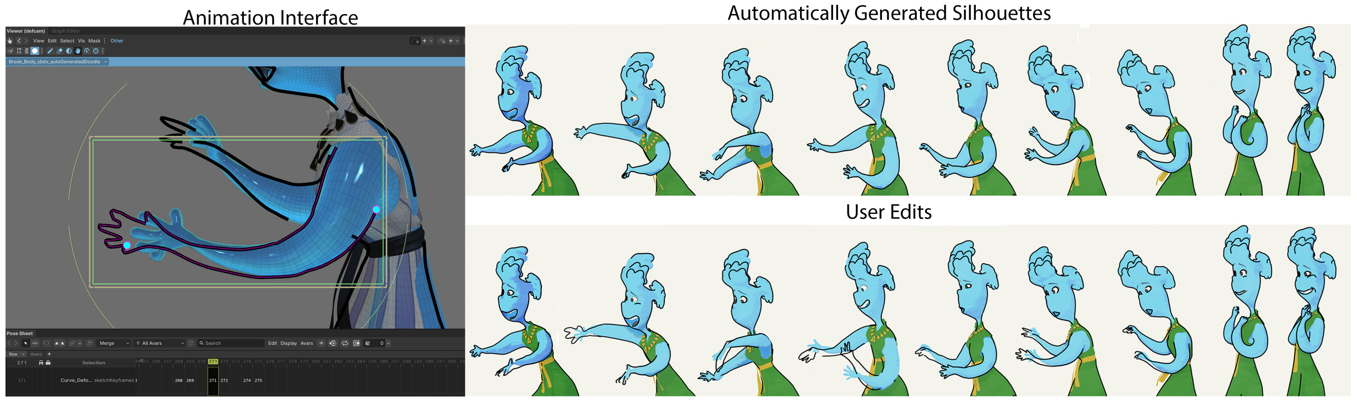

Figure 1: With our animation interface, users can edit the shape and opacity of automatically generated silhouette curves in a temporally consistent manner. In this case, a user changed the shape of the character's hand lines through editing and redrawing the curves to emphasize the movement. Towards the beginning and end frames, the user also added laugh lines under the character's eye and around the mouth. ©2023 Pixar

ABSTRACT

Linework on 3D animated characters is an important aspect of stylized looks for films. We present CurveCrafter, a system allowing animators to create new lines on 3D models and to edit the shape and opacity of silhouette curves. Our tools allow users to draw, redraw, erase, edit and retime user created curves. Silhouette curves can have their shape edited or reverted, and their opacity erased or revealed. Our algorithm for propagating edits over tracked silhouette curves ensures temporal consistency even as curves expand and merge. Five professional animators used our system to animate lines on three shots with different characters. Additionally, the effects lead from the short film *Pete* used our system to more easily recreate edits on a film shot. CurveCrafter was able to successfully enhance the resulting animations with additional linework.

KEYWORDS

Animation, Creativity support, Silhouette contours

ACM Reference Format:
Nora S Willett, Kurt Fleischer, Haldean Brown, Ilene L E, and Mark Meyer. 2023. CurveCrafter: A System for Animated Curve Manipulation. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. **https://doi.org/10.1145/3586183.3606792**

1 INTRODUCTION

When determining the look of a film, artists take into account multiple elements including color, texture, and lines. Line work in particular is important for defining the shape of characters and scene elements and directing the viewer's attention [14]. Multiple recent 3D animated films incorporate lines into their style. In *Paperman*, artists draw lines to define the silhouettes of characters as well as internal features [23]. For *Spiderman: Into the Spiderverse*, lines are used as detail elements for facial expressions and special effects [35]. For painterly looks, lines are useful for clarifying edges and directing the viewer's gaze. *The Mitchells vs The Machines* and *The Bad Guys* use colored silhouette lines to emphasize the character's clothing and hair [32, 34]. *Pete* adds black sketch lines around characters to offset them from the watercolor background [33]. In all of these examples, line work is integral to the final look and read of the characters.

To create these styles, artists used a variety of systems which run the gamut of 2D vs 3D and manual to automatic. In the 2D space with more manual interventions, *Paperman* used Meander [42]. Lines are drawn in 2D for a frame and then moved between frames using 3D motion data. In *The Mitchells vs The Machines*, silhouettes lines are automatically detected in image space on the edges of characters and rendered in a thick marker style with their color complementing the source location [15]. Another process is creating lines directly on the 3D models. The lines in *Spiderman* were drawn and rigged in 3D with the line locations initialized by machine learning [38]. *The Bad Guys* used a combination of rigged lines on characters, drawn lines on 3D sets, and image space detected silhouettes [8, 9]. In *Pete*, the black outlines are detected in 3D space on the models and then rendered using brush strokes which are composited on top of the watercolor rendered backgrounds.

For all of these types of lines, they are either manually drawn or automatically generated. When automatically generating lines on 3D models, the resulting curves may not be exactly what the artist intended for that frame. As a result, a system is needed to allow artists to add, edit or delete automatically generated 3D lines.

To investigate current workflows for fixing auto-generated lines on multiple frames, we interviewed the effects lead on *Pete* about their process (Figure 11). For each shot, 3D silhouette lines were generated on the characters and props. However, this method results in artifacts where lines can sometimes appear and disappear [10]. Since lines are individually generated at each frame, any fixes on one frame need to be manually performed again on subsequent frames. For fixes, the lines were brought into the Meander system [42] where they were edited. About 70% of the shots in the 6 minutes and 30 second film had manual fixes to the lines. With 8,066 frames, 30 to 40% of those frames were manually edited. The majority of those fixes (95%) were erasures and the remaining 5% was a combination of an erase and draw to make a line consistent across multiple frames. Most of the fixes were on the characters' faces (nose and jawline) since small changes in facial lines affect the expression the animator is trying to achieve.

We propose a novel UI, CurveCrafter, which allows users to draw, edit and erase new lines on 3D models. These lines are deformed with the model and interpolated between frames. For silhouette lines on characters, users can deform or erase the lines on one frame, and the changes will be propagated to nearby frames. Our workflow improves the experience and reduces the tedium of manually editing every frame to fix discrepancies in automatically generated silhouette lines.

To validate our system, we conducted a user study with five professional animators on three production shots. We also had the effects lead on *Pete* use our system to recreate some of the edits necessary for the final film. Our examples show comparisons of the automatically generated silhouettes with the users' edited versions.

Our contributions are:

- A UI with tools allowing users to edit the shape and opacity of 3D silhouette curves.
- An algorithm to propagate edits on silhouette curves between frames improving temporal consistency.
- An evaluation with five professional animators using our system to create line edits on three shots.
- Comparison of our system to Meander [42] using a shot from the short film *Pete*.

In the following sections, we will discuss related work and the challenges of editing automatically generated silhouette curves. After describing our UI and how it is implemented, we will share the findings from the user study and the animated results that were created. Finally, we use a shot from the short film *Pete* to compare our UI to Meander [42] and discuss some limitations and areas for future improvement.

2 RELATED WORK

We discuss previous work on sketch based interfaces, line editing interfaces, silhouette generation, and inbetweening.

2.1 Sketch Based Interfaces

Sketch based interfaces with interactions in 3D fall into two main categories: those that focus on AR/VR and those that work with 3D modeling. When sketching in AR, users can utilize a combination of 2D, 3D and mixed reality techniques [3, 31, 44]. For using sketching to build 3D models, some works are tailored towards novices [4, 28] while others focus more on interaction techniques involving pen and hand motions [29]. Most implementations convert 2D drawn strokes to 3D curves or geometry [19, 24, 41, 45]. Additionally in sketch based interfaces, there is a strong desire to stylize the strokes that are created on 3D models [11, 12, 26].

2.2 Line Editing Interfaces

One of the most popular programs for editing lines is Adobe Illustrator which includes a range of functionality for drawing, manipulating and erasing vector line drawings [2]. However, these operations are designed to perfect a single image and do not offer techniques for multiple frames.

Examples of 2D animation systems include ToonBoom's Harmony, TVPaint, and Disney's Meander. Harmony and TVPaint allow for a full suite of tools from drawing and editing lines to keyframing, stroke matching and inbetweening [17, 40]. Meander is a hybrid

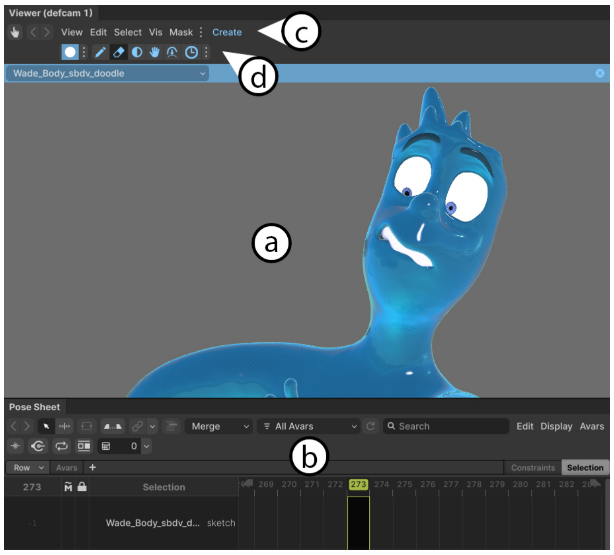

Figure 2: Interface for the curve editing system. (a) The 3D viewer. (b) The pose editor. (c) Menu to create silhouette curves. (d) In order from left to right: Draw Tool, Erase Tool, Reveal Tool, Edit Tool, Revert Edit Tool, Retiming Tool. ©2023 Pixar

of a 2D animation system with 3D input [42]. Artist-drawn curves from a starting frame are advected to subsequent frames using a vector field calculated from the motion of underlying 3D models. Then, the artist can choose curves on subsequent frames to edit and fix. Because curves are advected and edited on subsequent frames, there is a direct correspondence between curves on different frames enabling matches for inbetweening.

In 3D software, Blender has implemented the Grease pencil object which allows users to draw and edit 3D curves [1]. Ways to expand the Grease Pencil structure to allow for inbetweening have also been explored [22]. Other 3D drawing systems include Dreamworks' Squiggles which allows static curves to be drawn on 3D models [8, 9].

Our system works with 3D curves and allows for interactivity when drawing and editing curves. It also incorporates inbetweening, and propagates edits between frames for automatically generated silhouette curves. For the interpolation of silhouette curve edits, we are unaware of previous work in academia or industry that solves this problem.

2.3 Silhouette Generation

Before editing, silhouette curves must be detected over 3D objects. Some techniques generate these curves from a 2D rasterized image based on edge filters [5, 20, 25, 27]. Other techniques use the 3D geometry as input to create 3D curves [6]. A survey by Bénard and Hertzmann provides a more in depth discussion of different silhouette generation methods [10].

2.4 Inbetweening

Inbetweening is a challenging problem with two parts, stroke matching and interpolation. An early work by Reeves compares three different algorithms for inbetweening [36]. Kort *et al.*uses stroke chains for matching and artistic control of the interpolation timing [30]. A more specific case of similar keyframes was explored by Whited *et al.*[43]. If using vector graphics, Dalstein *et al.*introduce a new data structure to support merging, splitting, appearing and disappearing of strokes [16]. For cases that are slightly more complicated than only 2D, Rivers *et al.*explore inbetweening in the 2.5D space [37].

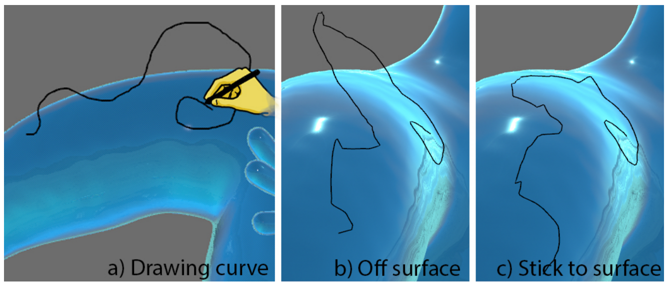

Figure 3: (a) Drawing a new curve with (b) the default projection or (c) projection to the surface. ©2023 Pixar

3 EDITING AUTO-GENERATED CURVES: CHALLENGES

When interpolating or propagating lines edits between frames, a critical component is stroke correspondences across frames. Stroke matching can be solved in a variety of ways. For strokes drawn at one frame and edited at others, there is an inherent correspondence. This method is the one that we use for matching. For strokes drawn at different frames, some algorithms attempt stroke matching but this process usually requires the changes in stroke shape to be minimal [43]. For curves automatically generated from 3D models, the curves are advected based on mesh movements in order to create correspondences between frames [7].

When adding linework to animation, it is very labor intensive for artists to manually draw all the silhouette and accent lines on a 3D character. Computing those lines automatically saves artists effort but traditionally limits the ability for control.

One challenge of editing automatically generated curves is their frame density. These curves are calculated and keyframed at every frame during the animation. In other systems, the curve are sparsely populated only on keyframes with intermediate frames interpolated. For our application of auto-generated curves, only edits and erasures have sparse keyframes whereas the curves being manipulated are keyed at every frame.

Due to these constraints, we propose a novel UI system which enables artists to edit automatically generated silhouette curves.

4 USER INTERFACE

Our system expands on UI elements native to traditional 3D animation software. We incorporate a pose sheet to display and edit keyframes and a 3D viewer to display 3D models and curves (Figure 2). The pose sheet shows the keyframes of different edits and curves, allowing the viewer to delete or retime curve animations (Figure 2b). The 3D viewer enables the user to manipulate the viewing angle and contains multiple predefined camera positions (Figure 2a). The options for interacting with curves on a model are located at the top of the viewer. Under the *Create* menu, one option allows users to enable editing on a selected model (Figure 2c). The other option automatically creates silhouette curves for the model and viewer's camera position given the current frame range. A title bar contains the tools for editing the curves (Figure 2d). The **Draw Tool** allows users to create curves, or replace a curve by redrawing its shape (Figure 3a and 4a), while the **Erase Tool** removes curves (Figure 4c and 5e). The **Edit Tool** enables a manipulative widget, similar to the one available in Meander [42], for editing the shape and position of the curves. Arcs on the side allow for rotation while the bounding box controls transformation and scaling. Clicking directly on the curve introduces a knot which enables more controlled deformations (Figure 4b and 5b).

To edit automatically generated silhouette curves, two additional tools are introduced. With the **Reveal Tool**, any invisible curve parts are shown with a pink line so they are easily detectable to be brushed over and revealed (Figure 5f). To undo a deformation of a curve, the user brushes over the curve with the **Revert Edit Tool** which slowly pushes the curve shape back to the original silhouette curve (Figure 5d).

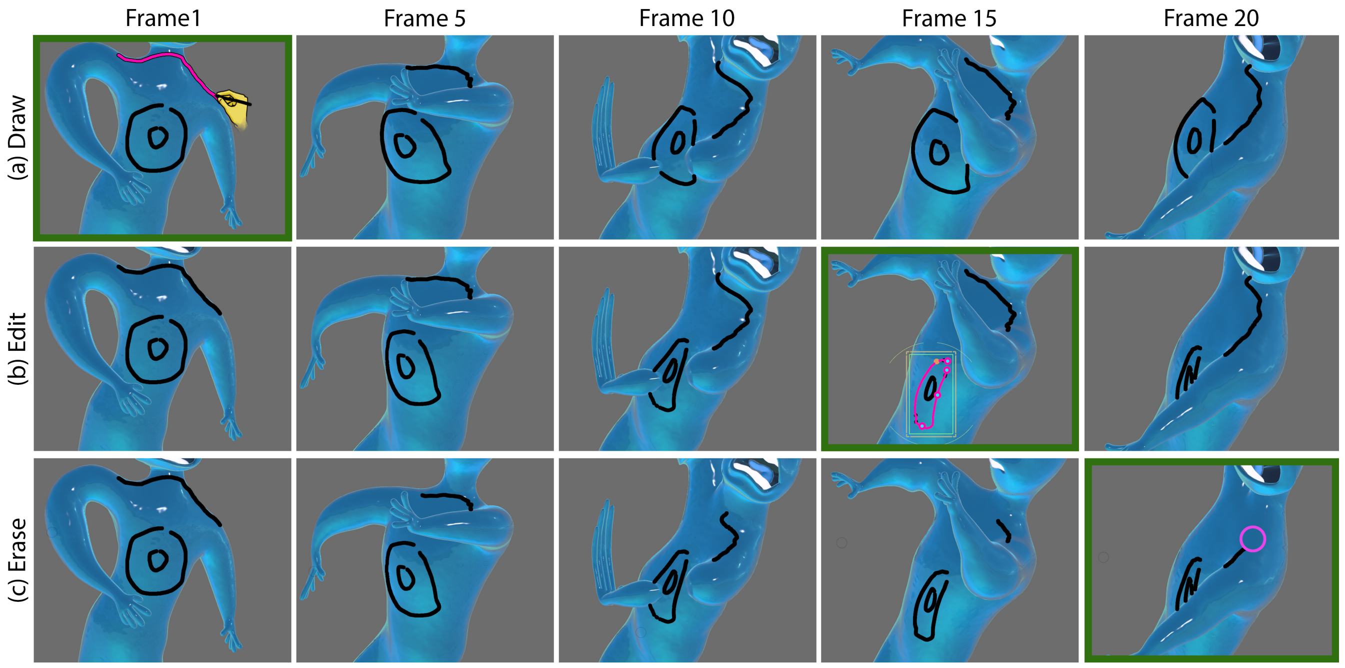

Figure 4: User created curves (a) The user draws curves at frame 1 and they are deformed with the mesh on subsequent frames. (b) With the edit tool, the user changes the curve shape at frame 15. Notice how the interpolated curves at frames 5 and 10 differ from the initial curves in (a). (c) The user erases part of a curve at frame 20. Watch the curve shrink in frames 5, 10, and 15. ©2023 Pixar

Other features include the **Retiming Tool** which, in conjunction with the pose sheet (Figure 2b), allows users to change the keyframes for the selected curves.

See the accompanying video for each tool's interactions.

5 IMPLEMENTATION

To store the curve data, we create a hierarchy consisting of sketches, curves and animated points. Sketches are keyed at individual frames and contain references to curves. A curve can belong to multiple sketches and stores the connectivity of *animated points*, allowing the positions along a curve to change between sketches. Curves also store an id to create a correspondence for interpolation. An *animated point* can belong to multiple curves and stores a mapping of sketches to position data. At runtime, the display position of the *animated point* is calculated from the deformed mesh based on a face id, UV coordinate and offset. With this structure, sketches correspond to a single frame, while curves and *animated points* can span multiple frames allowing points to be consistent across curve topology and sketches at different frames.

For all the interactions, the user's input is in 2D image space (Figure 3a) while the stored data is in 3D relative to the mesh. The 3D positions are calculated by casting rays from the 2D image space point to intersect with the selected 3D mesh. If the ray interests the mesh, the 3D position is stored using the mesh face id, UV coordinates and offset in relation to the face's normal (in this case, a vector close to zero length). If no intersection occurs, the user can choose two projection options.

In one option, **Off surface** projection (the default), the point's distance from the camera is copied from the closest point to the 2D mesh's silhouette (Figure 3b). Distances are smoothed along the curve to avoid sharp jumps when switching between closest mesh positions (i.e., from being close in 2D to a foot and then to a hand). With the camera distance, the 3D world point is calculated and the closest face on the 3D mesh is stored along with the UV and vector offset.

For the other projection option, **Stick to surface**, the calculation of the 3D world point is the same as above. However, to ensure that the points stick to the mesh surface, the offset is set to zero, forcing the point to lie on the closest mesh face (Figure 3c).

Our interface, which was implemented inside of Pixar's Presto animation system [39], allows animators to interact with user created curves and automatically generated curves using the same UI interactions. However, on the backend, we store and interpret the data in slightly different ways.

5.1 User Created Curves

To create a new curve, the Draw Tool generates curves with new points for each drawn stroke. The type of projection (Off surface or Stick to surface) used for each curve is set at this stage (Figure 3). All curves drawn at a frame are contained within a single keyframed sketch. Drawn curves at different frames are not matched for interpolation.

When editing a curve, the user can **redraw** or **deform** it. For redrawing, the old curve is replaced with the new drawn shape. For

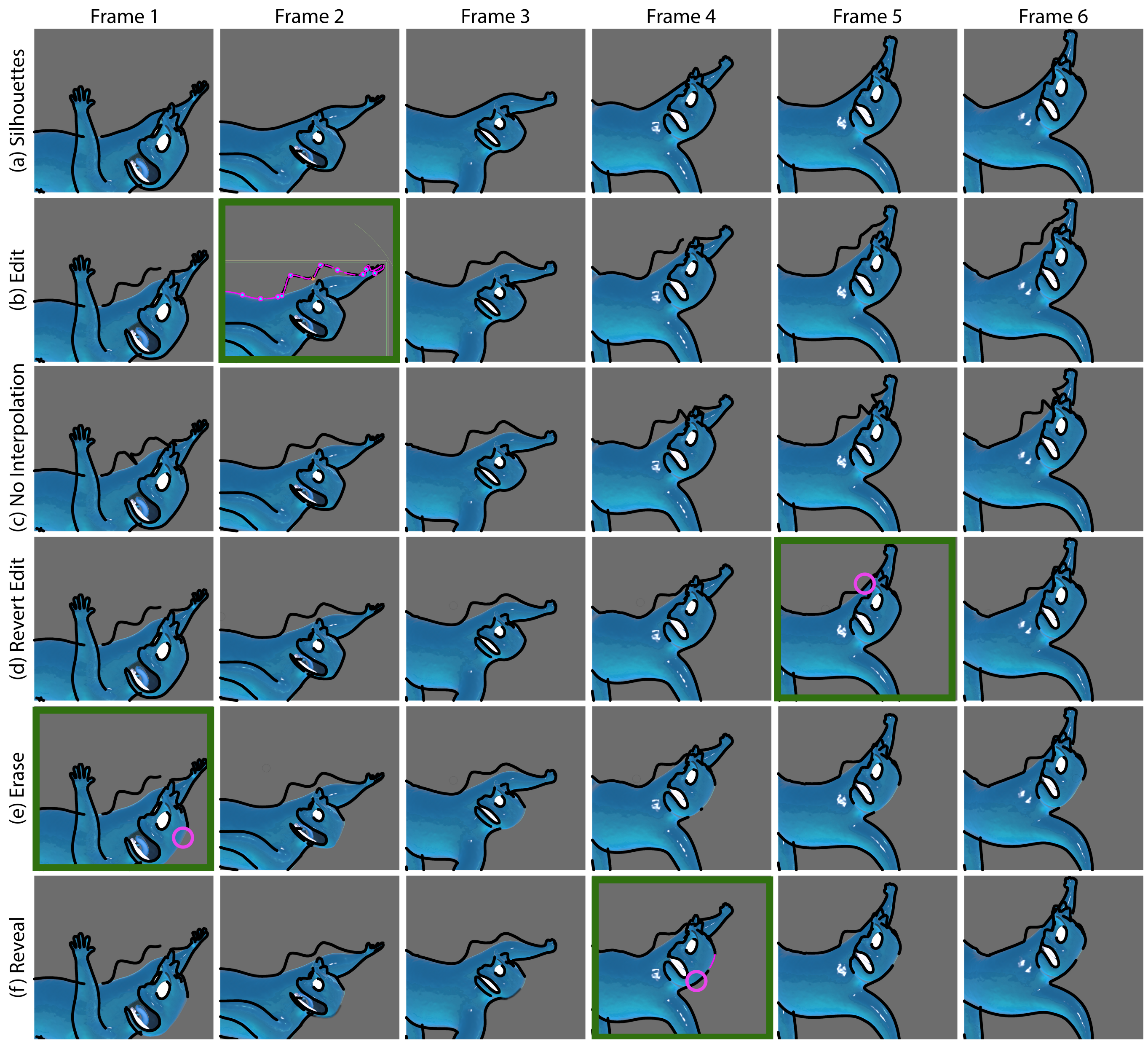

Figure 5: Auto-generated curves (a) The automatically generated silhouette lines. (b) The user edits the shape of a curve on frame 2 and the edit is geometrically and temporally interpolated on subsequent frames. (c) If our geometric interpolation method is not used, the curve has lots of gaps and jaggies (frames 1, 4, 5, 6). (d) The user can brush over areas to revert some edits such as removing a bump on frame 5. The offsets are interpolated on frames 3 and 4 to shrink the bump over time. (e) With the erase brush, the user can make some curves disappear on frame 1. (f) The reveal brush shows which areas can be made visible in pink and lets the user brush over them to make them reappear in frame 4. ©2023 Pixar

deforming the old curve, the edit widget defines the deformation based on the rotation, transformation or scaling. If knots are used to change the shape, we use a Catmull-Rom spline [13] to calculate offsets for a single selected curve and as-rigid-as possible curve editing to calculate the new shape of multiple selected curves [21]. After deformation, a new curve with new points (not necessarily the same number as before) is created at the frame (Figure 4b). However, the new curve's interpolation id matches the old curve, creating a correspondence for inbetweening.

To erase strokes, we search for points within a given brush radius in 2D image space. If the points erased are at the ends of the curve, we shorten the curve by deleting those points (Figure 4c). If the

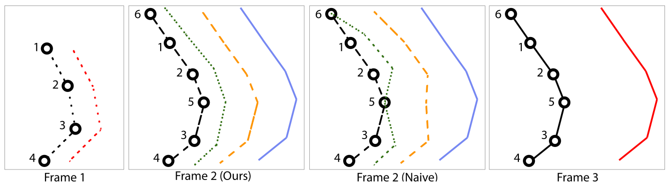

Figure 6: Expanding automatically generated curves: The original curves are represented in black with labeled animated points. The user sets keyframed offsets in red for frame 1 and 3 by editing the curves. When expanding the curve on frame 2, our method *geometrically* interpolates offsets values from both frames (green and blue curves) and then *temporally* interpolates for the final orange curve. If the keyframed values are naively copied, the result is not temporally coherent.

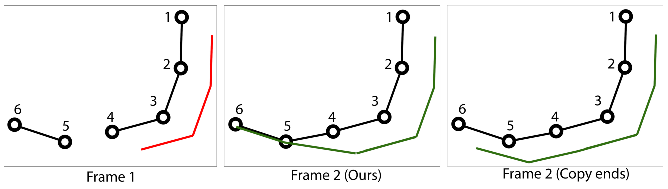

Figure 7: Merging automatically generated curves: The original curves are represented in black with keyframed offsets in red. When merging curves, directly copying the keyframes and expanding the values along the curve ends results in a jump between frames. Since points 5 and 6 belong to a separate curve on frame 1, during our *geometric* interpolation method, the offset value from point 4 is not copied to points 5 and 6.

erased points are in the middle, the existing curve is shortened from one end and new curves are created for the other visible parts. If a curve is completely erased at a frame, a curve with zero points and a matching interpolation id is saved to ensure correct inbetweening.

To calculate which curves to display at any given frame, each unique interpolation id is considered separately and the closest keyframes to the display frame are recorded. If there is a keyframe at the display frame or only one closest keyframe, either before or after the display frame, that curve is displayed from the deformation of the mesh. If the display frame is between two keyframes, each curve is deformed by the display frame's mesh, sampled to contain the same number of points (we use the maximum of the two curves' point counts), and then points are linearly interpolated. If one curve has zero points meaning it was erased completely, the other curve is displayed. See Figure 4 for interpolation results with various edits and erasures.

5.2 Auto-generated Curves

Our automatically generated curves, represented as 3D polylines, are extracted from the surface mesh using a combination of techniques informed by Bénard and Hertzmann [10]. We generate silhouette curves by tracing the zero levelset of the dot product between normals at mesh vertices and the view direction [20] (Figure 5a). Once visible curves are calculated at each frame, we need to determine where points from the current frame move to in the next one. This tracking allows for an *animated point* at the first frame to change positions and curve connectivity over time ensuring deformations and visibility edits are consistent. To create correspondences between different frames, we advect the curves in 3D from the previous frame based on the mesh movement between frames [7]. When the corresponding projected 2D image space curves are close together, we update the previous points' positions to lie on the current curve, using nearest neighbor and tangent similarity. For any parts of current curves that are not covered by curves from the previous frame, we introduce new points to display those areas.

When editing the curves, we keyframe a *deformer* that stores the ids of the edited animated points and the global offsets in 3D (Figure 5b). Next, we need to handle how a deformation is interpolated across multiple frames for smooth temporally coherent results. As curves expand, contract and merge, the number of points changes. Therefore, points may only exist on one of the keyframes and directly interpolating offsets for each point will result in jaggies and divots where there is no offset to interpolate (Figure 5c). Our innovation is the proposal of an algorithm that produces pleasing temporal results which align with what animators would expect. Our system deals with the cases of expansion and merging of curves slightly differently.

The first case to consider is a curve expanding (Figure 6). If a point has an offset keyframed at the current frame, that value is used. For all other points along a curve, we store the previous and next keyframes and their offsets. Then, we need to calculate the offset values for each point at the current frame. Using the curve topology for the current frame, we *geometrically* interpolate along the curve the offset values separately for the previous frame (Figure 6, Frame 2 (Ours) green) and then the next frame (Figure 6, Frame 2 (Ours) blue). For *geometric* interpolation, points at the ends of curves are treated differently than those in the middle. For points in the middle, the offsets are linearly interpolated. For points on the end, the offset values of any new points are copied from the last keyframed point on the curve. Then, we *temporally* interpolate the offset values resulting in the final temporally consistent displayed curve (Figure 6, Frame 2 (Ours) orange).

In the second case of curves merging, the end points belonging to a curve at any frame which does not share topology with the keyframed points are not *geometrically* interpolated and have zero offset. This method ensures that a separate curve with no deformations stays that way even if it merges with a curve that has deformations applied. Otherwise, a jump will occur at the merged frame as the offset is extended to the "new" points of the curve (Figure 7).

When using the **Revert Edit Tool**, a brush stroke shrinks the affected offset vectors by 0.1 until there is zero offset (Figure 5d).

To erase auto-generated points, we keyframe the brushed over points' opacities at the current frame to zero. Additionally, the **Reveal Tool** keyframes the opacities to one. Similarly to how offsets are calculated for display, the opacities are also *geometrically* and *temporally* interpolated (Figure 5e,f).

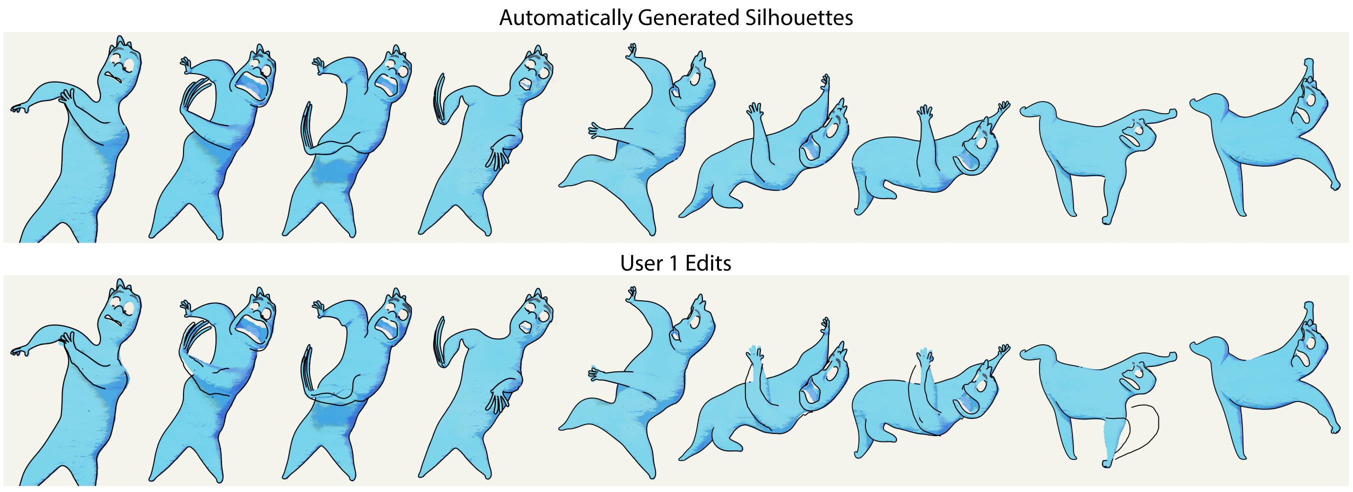

Figure 8: In shot 1, the user changed the shape of the arm's silhouette. ©2023 Pixar

6 USER STUDY

To test our UI in a production setting, we gathered feedback from five professional 3D animators with an average of 15.8 years (6.5 std dev) of experience in 3D and 6 years (6.4 std dev) of experience in 2D. After demonstrating all the UI tools including interacting with user created and automatically generated curves, the participants were presented with a different character to experiment on. They were encouraged to use the tools to add decorative elements which would enhance the animation as well as edit the silhouette curves. All silhouette curves were generated, tracked and loaded before the animators worked on their shots. Overall, the demonstration session lasted 20 minutes while animators spent an average of 96 minutes (52 std dev) working on their shot. Each participant created a single result on one of three characters.

In general, users found our tool exciting to use and intuitive to draw and edit directly on the model. They were pleased with the animations created and envisioned using this tool for drawing details on cloth and faces, adding effects, and fixing silhouette lines around the face. Animators could indicate folds in the cloth that could be passed to the simulation department. They appreciated being able to add details to the face to refine and push the appeal of the characters' expressions. Other uses include creating effects such as cloud puffs, surprise marks, or bubbles.

Users liked the speed and ease of the draw and redraw functionality and used it extensively to refine the shapes of curves. Workflows included drawing a curve multiple times until the exact shape was achieved or redrawing an interpolated deformed curve on a different frame to better match a desired shape. When using the erase tool, animators requested that erased user generated curves behave more like erased silhouette curves so that the reveal tool can work on both types. This change would allow animators to erase parts or all of a curve at a frame and then make the exact curve reappear later. Additionally, animators wanted control over the brush size.

For the retiming tool, animators wanted the ability to see keyframes in a spline editor as well as the pose sheet. This additional view would allow more flexibility and transparency into how keyframes are held and interpolated (using splines controlled by the animator instead of our default linear option).

Another general request from one user was a more streamlined selection of curves. Currently, each tool that requires a curve to be selected (redraw, edit, retiming), has its own selection inside the tool before the operation can take place. With a single persistent selection tool, subsequent actions would know which curve to act on.

When editing the silhouette curves, the animators used a combination of the edit tool to change the shape of the curves and the erase and draw tool to completely redraw the curves. Many users expressed the desire for the redraw functionality of the draw tool to also work on silhouette curves instead of just user drawn curves. When using the revert tool, users had several suggestions. They wanted to change the strength and size of the brush in order to have more control over how it returns curves to the original silhouette. One user also suggested a hotkey to quickly snap a curve back to its original shape. Two users mentioned that having some indicator of which curves are deformed and how much would be nice. When using the reveal tool, three users mentioned combining this tool with the erase tool to allow users to directly set the opacity of curves.

All animators suggested the addition of a smoothing tool. This option could be used to brush over drawn or edited curves to remove small bumps or jaggies.

Two animators mentioned that they envisioned using the silhouette edits to change the underlying geometry. They would use the drawn and edited curves to quickly pose the 3D character similar to a process used on *Inside Out* [18].

7 RESULTS

Users worked on three shots featuring a single different character in each. Refer to Table 1 for the time that each user spent on their shot and the percentage of frames that were edited. Please see the accompanying video for all user created results.

The first shot, lasting 138 frames, has a character walking and then turning around in surprise (Figure 8). The character has 26,892

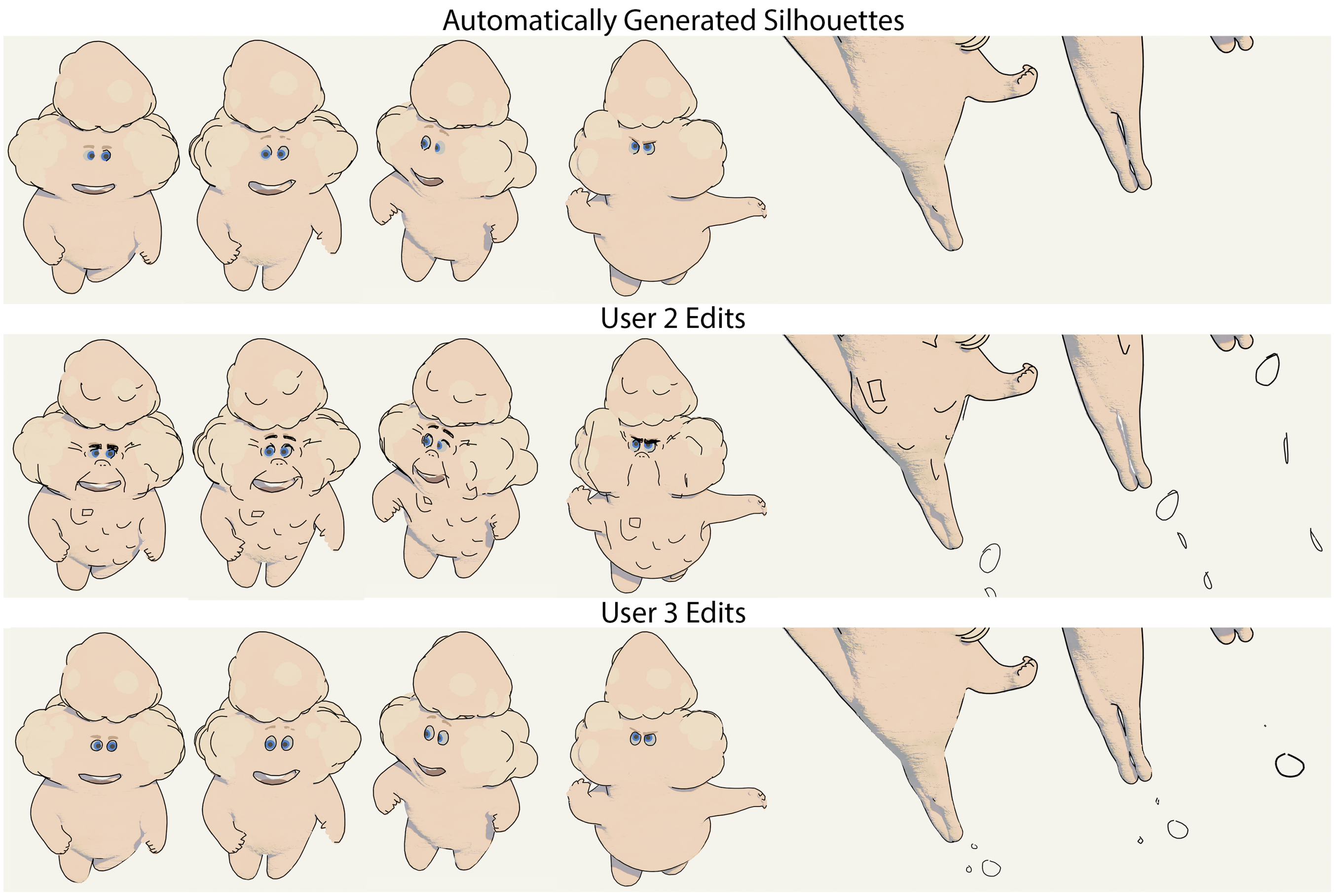

Figure 9: In shot 2, both users edited the eye shapes and added puffs at the back as the character flies away. ©2023 Pixar

faces and it took 57 minutes and 29 seconds to generate and track the silhouettes (average 24 seconds per frame). The frames have an average of 25 curves with a standard deviation of 5.19. The animator used our system to refine the curves indicating the arm shapes as the character bends backwards. They achieved their edits in two ways. For some frames, they used the edit to tool change the shape of the silhouette arm curves. In another instance, they erased the arms' silhouettes and redrew the curves themselves.

In the second shot (63 frames), the character looks around, puffs up and then flies away (Figure 9). The character has 55,264 faces and silhouette generation and tracking took 78 minutes and 49 seconds (average 75 seconds per frame). The frames have an average of 51 curves with a standard deviation of 3.99. Both animators editing this shot focused on the character's face. One drew extra lines around the eyes, eyebrows and cheeks. Another mainly focused on the lines around the eyes. They erased the silhouette generated eye lines on most frames and redrew them. Both animators added bubbles at the end of the character as they fly off the screen.

In the final third shot, the character hesitantly reaches out and back (110 frames) (Figure 10). The character and dress have 79,503 faces and it took 93 minutes and 23 seconds to generate and track the silhouettes (average 50 seconds per frame). The frames have an average of 47 curves with a standard deviation of 4.54. One animator pulled the silhouette at the back of the character's dress and head out in order to create a bubble that breaks off and floats away. The other animator focused on the hand shapes and added details around the face (Figure 1). They drew extra wrinkle lines under the eyes and in the corner of the mouth, using the tools to make the lines appear, grow, shrink and then disappear. To change the hand shapes to overshoot the motion, they used a combination of editing the silhouette lines and redrawing the lines for more control around the finger shapes.

	Shot 1	**Shot 2**		**Shot 3**	
Users	1	2	3	4	5
Time spent on task (minutes)	50	90	150	40	150
User created keyframes	9%	22%	62%	12%	31%
Unique curves	10	63	8	4	12
Silhouette shape keyframes	6%	3%	5%	14%	23%
Silhouette opacity keyframes	4%	11%	73%	0%	16%

Table 1: Summary of users' interactions with our system on a variety of shots.

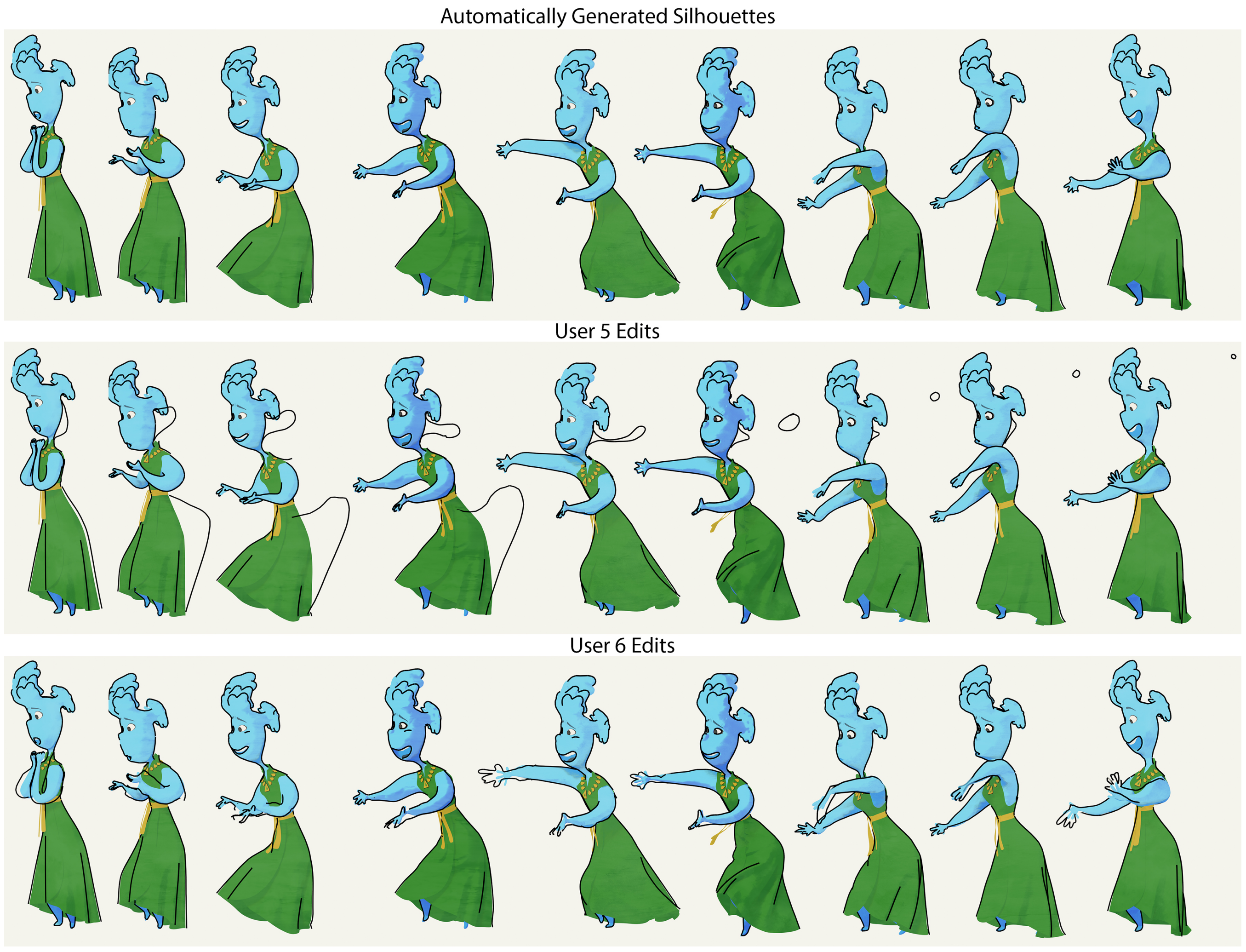

Figure 10: In shot 3, user 5 makes a bubble break away from the character's back and head silhouette. User 6 changes the hand and arm silhouette lines to emphasize the reaching motion. ©2023 Pixar

8 USE ON A SHORT FILM

To compare our system directly to existing workflows, we asked the effects lead on *Pete* to redo a shot from the film using our tool (Figure 11). The production pipeline exported the animation to Houdini to generate silhouette curves and then, the curves were transfered to the Meander system [42] where each frame was edited and rendered. Using our UI, the silhouette curves are created and edited inside of the animation software. They are then exported to Houdini for additional stylization processing and then to Meander for rendering.

To test the effectiveness of our system, the effects lead chose a shot which required significant erasures and edits of the silhouette lines. The shot (123 frames, 57,880 faces, average 33 silhouette curves per frame) features a close up of Pete slightly nodding while smiling and blinking. The fixes to the silhouette lines include reducing flickering in the instances of a hat line gap and the armpit lines. Other fixes focused on closing the gap on the chin, and removing a part of the left ear line to fix the occlusion with some hair strands.

To achieve all of these fixes in Meander, all 123 frames needed editing to erase the hat line gap and ear overlap using the erase brush. The armpit lines where not removed due to the tedious nature of selecting and deleting them on every frame. While Meander does offer the capability to add lines per frame by drawing with a pen, this method was only used rarely on single frames. Due to the stylization method of the lines, which included procedurally adding line wobble and pressure, recreating a consistent technique when drawing was nearly impossible to ensure temporal coherence.

For both Meander and CurveCrafter, the effects lead spent 40 minutes making changes to the shot. However, with our more powerful tool, the edits were less tedious and allowed the effects lead to accomplish more, such as erasing the armpits and drawing lines on the hat and chin. In both systems, the majority of the time was spent flipping through frames to see how the edits look

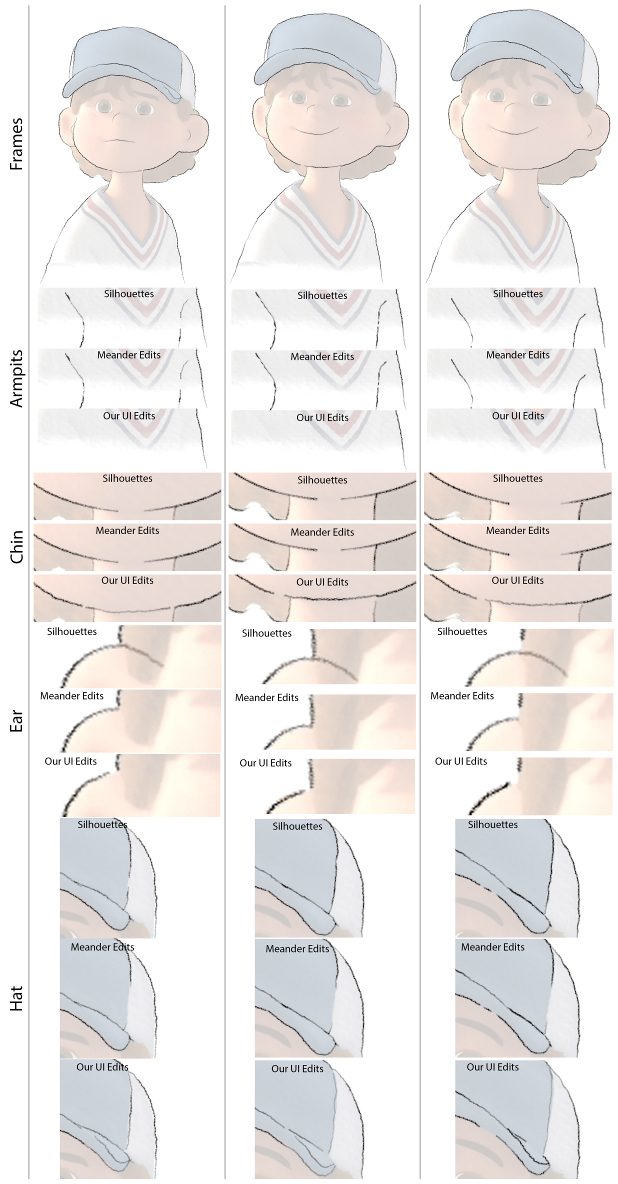

Figure 11: In a shot from *Pete*, we compare the edits from our UI to the original silhouettes and the edits done in Meander. For the armpits, our UI was used to erase the lines. However, those lines were not erased in the original film using Meander because the process was too tedious. On the character's chin, our system was used to draw a line closing the gap. Drawing a line to close the chin gap in Meander was not possible due to difficulties matching the pen pressure over multiple frames. For the ear line, our UI was able to erase it with one keyframe while the effects lead had to edit every keyframe to erase the line with Meander. For the hat, Meander was used to erase a portion to remove some flickering. With our system, the hat line was erased and then redrawn. Additionally, an extra line was added at the hat brim. ©2023 Bret Parker

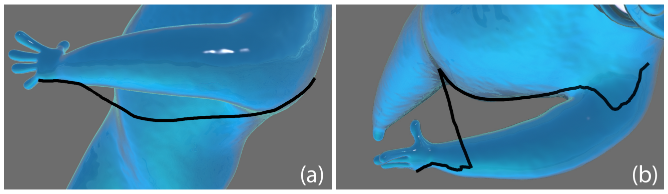

Figure 12: Limitation (a) Drawing a curve to accentuate the curve of the arm. (b) The curve is projected onto the belly which will result in unpleasant deformations on subsequent frames. ©2023 Pixar

in motion since neither system played back in realtime. Another benefit is that with less keyframes in our system, future edits to curves and their timing are easier to accomplish.

When using CurveCrafter, 43% less keyframes were need for erasing curves. Erasing the armpit lines required 70 keyframes while fixing the ear overlap only required a single keyframe. The large difference in keyframes needed for the various areas is due to the calculated line tracking. The armpit lines are not very temporally consistent since they reside in a crease area, and hence tracking is difficult between frames as small line bits appear and disappear. The ear line is very consistent across the whole shot and hence the tracking is able to easily propagate edits. To fix the flickering in the hat gap, a part was erased (10 keyframes) and two new lines were drawn on a single keyframe, one in the hat top and another along the bill. To close the chin gap, three keyframes were used to erase the end parts of the lines to make them more consistent. Then, a new line was drawn on one frame and edited on another to fill in the gap. Because our curves are drawn and saved in 3D, they can be fed into the procedural process which determines the stylized line wobble and pressure characteristics. This option enables the easy addition of new silhouette lines where they were previously missing. The effects lead commented that "it is easy to add lines which is super nice."

In summary, our UI and algorithm for propagating opacity and edits of silhouette curves produced significantly less keyframes for erasing curves and enables the addition of temporally consistent curves which was not previously possible with the given style in Meander.

Please see the accompanying video for the shot and its comparisons.

9 LIMITATIONS AND FUTURE WORK

Through interacting with our system, users encountered areas for future development.

When editing silhouettes to make a temporally consistent change, the number of keyframes is heavily dependent on the quality of the silhouette curve correspondences over time. One user struggled with erasing all the lines around the eyes, having to touch 73% of the frames. The eyes were a particularly challenging example for tracking due to the silhouette lines disappearing in some parts when the eyeball mesh obscures the eye socket silhouette. Higher quality silhouette detection and curve tracking would allow for less clean up work for animators.

Another limitation of the current system is that the silhouettes and their subsequent user edits are tied to the character's animated pose. If animators change the pose, new silhouette curves with different tracking information would need to be computed. Without a way to automatically transfer the user edits, all of their work would have to be redone. Smoothing out the workflow between animating a character's pose, generating silhouettes in real time, and keeping consistent user edits would greatly improve the system's interaction and conform to animators' desired workflow.

Animators encountered some struggles when projecting the curves onto the surface and having them deform. One instance is when drawing the silhouette of an arm crossed over the body (Figure 12a). When projecting the drawn curve, parts of the line project to the arm while others stick to the stomach area (Figure 12b). When the curve is deformed on subsequent frames, the resulting shapes are not what the animator intends and redraws are required at every frame.

10 CONCLUSION

We have presented CurveCrafter, a UI which allows animators to edit automatically generated silhouette curves while ensuring temporal consistency of those edits between frames. Additionally, users can draw, edit and erase their own 3D created curves which deform and interpolate over the 3D animation. With these features, CurveCrafter gives direct and easy control back to animators over the stylized lines in films.

REFERENCES

[1] Grease pencil introduction. *Blender Docs* (2022).
[2] Adobe. Illustator, 2022.
[3] Arora, R., Habib Kazi, R., Grossman, T., Fitzmaurice, G., and Singh, K. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), 1–15.
[4] Bae, S.-H., Balakrishnan, R., and Singh, K. Everybodylovessketch: 3d sketching for a broader audience. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (2009), 59–68.
[5] Ben-Zvi, N., Bento, J., Mahler, M., Hodgins, J., and Shamir, A. Line-drawing video stylization. In *Computer Graphics Forum*, vol. 35, Wiley Online Library (2016), 18–32.
[6] Bénard, P., Hertzmann, A., and Kass, M. Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics 33*, 2 (2014), 1–21.
[7] Bénard, P., Jingwan, L., Cole, F., Finkelstein, A., and Thollot, J. Active strokes: Coherent line stylization for animated 3d models. In *NPAR 2012-10th International Symposium on Non-photorealistic Animation and Rendering*, ACM (2012), 37–46.
[8] Budsberg, J., Valle, P., and de Guzman, P. Building an illustrated world in the bad guys. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks* (2022), 1–2.
[9] Budsberg, J., Valle, P., Sans, J., Costello, M., Augello, N., and de Guzman, P. Graphic 2d-inspired characters in the bad guys. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks* (2022), 1–2.
[10] Bénard, P., and Hertzmann, A. Line drawings from 3d models: A tutorial. *Foundations and Trends® in Computer Graphics and Vision 11*, 1-2 (2019), 1–159.
[11] Cardona, L., and Saito, S. Hybrid-space localized stylization method for view-dependent lines extracted from 3d models. In *NPAR@ Expressive* (2015), 79–89.
[12] Cardona, L., and Saito, S. Temporally coherent and artistically intended stylization of feature lines extracted from 3d models. In *Computer Graphics Forum*, vol. 35, Wiley Online Library (2016), 137–146.
[13] Catmull, E., and Rom, R. A class of local interpolating splines. In *Computer aided geometric design*. Elsevier, 1974, 317–326.
[14] Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H. S., Finkelstein, A., Funkhouser, T., and Rusinkiewicz, S. Where do people draw lines? In *ACM SIGGRAPH 2008 papers*. 2008, 1–11.
[15] Conference, S. The handmade look of 'the mitchells vs. the machines'. *ACM Siggraph Blog* (2021).
[16] Dalstein, B., Ronfard, R., and van de Panne, M. Vector graphics animation with time-varying topology. *ACM Trans. Graph. 34*, 4 (July 2015).
[17] Développement, T. Tvpaint animation.
[18] Fleischer, K., Isaacs, P., Parker, B., Haux, B., Shen, S., Krishna, V., Shen, C., Butts, A., Price, J., Hahn, T., et al. Silhouette sketching on "inside out". In *ACM SIGGRAPH 2015 Talks*. 2015, 1–1.
[19] Grimm, C., and Joshi, P. Just draw it! a 3d sketching system.
[20] Hertzmann, A., and Zorin, D. Illustrating smooth surfaces. In *ACM SIGGRAPH* (2000), 517–526.
[21] Igarashi, T., Moscovich, T., and Hughes, J. F. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG) 24*, 3 (2005), 1134–1141.
[22] Ilene, E., Willett, N. S., and Finkelstein, A. 2.5 d simulated keyframe animation in blender. In *34th Annual ACM Symposium on User Interface Software and Technology, UIST 2021*, Association for Computing Machinery, Inc (2021), 35–36.
[23] Kahrs, J. Paperman, 2012. Walt Disney Animation Studios.
[24] Kallio, K. 3d6b editor: projective 3d sketching with line-based rendering.
[25] Kalnins, R. D., Davidson, P. L., Markosian, L., and Finkelstein, A. Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (2003), 856–861.
[26] Kalnins, R. D., Markosian, L., Meier, B. J., Kowalski, M. A., Lee, J. C., Davidson, P. L., Webb, M., Hughes, J. F., and Finkelstein, A. Wysiwyg npr: Drawing strokes directly on 3d models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), 755–762.
[27] Karsch, K., and Hart, J. C. Snaxels on a plane. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2011), 35–42.
[28] Kazi, R. H., Grossman, T., Cheong, H., Hashemi, A., and Fitzmaurice, G. W. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *UIST*, vol. 14 (2017), 401–414.
[29] Kim, Y., An, S.-G., Lee, J. H., and Bae, S.-H. Agile 3d sketching with air scaffolding. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), 1–12.
[30] Kort, A. Computer aided inbetweening. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), 125–132.
[31] Kwan, K. C., and Fu, H. Mobi3dsketch: 3d sketching in mobile ar. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), 1–11.
[32] Michael Rianda, J. R. The mitchells vs the machines, 2021. Sony Pictures Entertainment.
[33] Parker, B. Pete, 2022.
[34] Perifel, P. The bad guys, 2022. DreamWorks Animation.
[35] Persichetti, B., Ramsey, P., and Rothman, R. Spider-man: Into the spider-verse, 2018. Sony Pictures Entertainment.
[36] Reeves, W. T. Inbetweening for computer animation utilizing moving point constraints. *ACM SIGGRAPH Computer Graphics 15*, 3 (1981), 263–269.
[37] Rivers, A., Igarashi, T., and Durand, F. 2.5 d cartoon models. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 1–7.
[38] Seymour, M. Ink lines and machine learning. *fxguide* (2019).
[39] Studios, P. A. Presto, 2023.
[40] ToonBoom. Harmony21, 2021.
[41] Tsang, S., Balakrishnan, R., Singh, K., and Ranjan, A. A suggestive interface for image guided 3d sketching. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (2004), 591–598.
[42] Whited, B., Daniels, E., Kaschalk, M., Osborne, P., and Odermatt, K. Computer-assisted animation of line and paint in disney's paperman. In *ACM SIGGRAPH 2012 Talks*. 2012, 1–1.
[43] Whited, B., Noris, G., Simmons, M., Sumner, R. W., Gross, M., and Rossignac, J. Betweenit: An interactive tool for tight inbetweening. In *Computer Graphics Forum*, vol. 29, Wiley Online Library (2010), 605–614.
[44] Xin, M., Sharlin, E., and Sousa, M. C. Napkin sketch: handheld mixed reality 3d sketching. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (2008), 223–226.
[45] Xu, P., Fu, H., Zheng, Y., Singh, K., Huang, H., and Tai, C.-L. Model-guided 3d sketching. *IEEE Transactions on Visualization and Computer Graphics 25*, 10 (2018), 2927–2939.