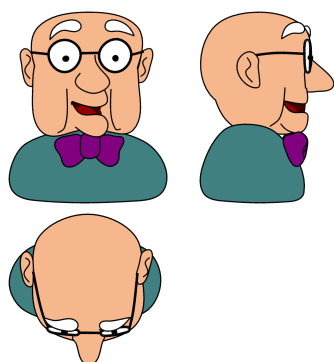# 2.5D Cartoon Models
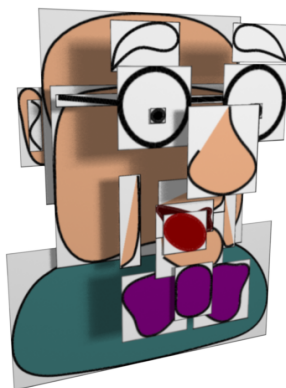
Alec Rivers
MIT CSAIL

Takeo Igarashi
The University of Tokyo
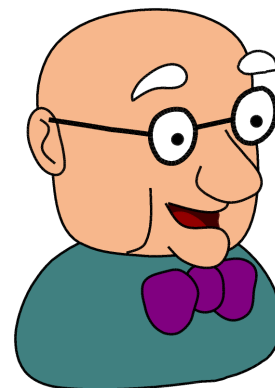
Frédo Durand
MIT CSAIL

(a)   (b)   (c)

Figure 1: A 2.5D Cartoon: *We take vector art drawings of a cartoon from different views (a) and use them to automatically generate a 2.5D cartoon (b), which associates each stroke with a 3D position. The 2.5D cartoon can then be used to simulate a rotation in 3D and generate a rendering of the cartoon in a novel view (c).*

Abstract

We present a way to bring cartoon objects and characters into the third dimension, by giving them the ability to rotate and be viewed from any angle. We show how 2D vector art drawings of a cartoon from different views can be used to generate a novel structure, the 2.5D cartoon model, which can be used to simulate 3D rotations and generate plausible renderings of the cartoon from any view. 2.5D cartoon models are easier to create than a full 3D model, and retain the 2D nature of hand-drawn vector art, supporting a wide range of stylizations that need not correspond to any real 3D shape.

Keywords: non-photorealistic rendering, cartoons, vector art, billboards, animation, interpolation

1 Introduction

The ability to rotate a cartoon and view it from any angle has wide-ranging applications, both in aiding animation and in enabling cartoon objects to be placed in an interactive 3D environment in which the user controls the viewpoint. Previously, this has been achieved by constructing a 3D model of the cartoon and rendering it in a non-photorealistic way so as to resemble a cartoon. However, generating a 3D model is time-consuming, and many stylistic elements of 2D drawings cannot be adequately reproduced in a 3D model,

as the 2D appearance intended may not correspond to any real 3D shape.

We propose a way to use 2D vector art drawings of a cartoon from different angles to generate a new type of structure, the 2.5D cartoon model, visualized in Figure 1 (b). This structure associates each stroke of a cartoon with a single 3D position, and is able to generate plausible renderings of the cartoon in new views by translating the strokes' positions in 3D, while interpolating their shapes in 2D. We have found that this simple model structure allows surprisingly believable rotations of cartoons, despite having no explicit 3D polygonal mesh of the object, and typically only requires three or four defined views of the cartoon to be able to generate plausible renderings of the cartoon in any orientation. The result is a cartoon that retains the 2D, hand-drawn nature of the input vector art, while supporting full 3D rotation.

2 Related Work

Cartoon-style drawing and animation have over time developed a rich language of non-photorealistic stylizations (see, e.g., [Johnston and Thomas 1981; Blair 1994]). Much work has been done that seeks to introduce these stylizations to computer rendering (see [Gooch and Gooch 2001] for a survey). Non-photorealistic rendering techniques have been proposed to make a 3D model resemble a hand-drawn cartoon, such as cel-shaded lighting [Decaudin 1996] or exaggerated, 2D silhouettes [Northrup and Markosian 2000; Kalnins et al. 2002].

However, rendering from a single 3D model cannot account for cartoons that have mutually inconsistent appearances in different views – for example, no matter the viewing angle, Bugs Bunny's ears are always facing the camera. View-dependent geometry [Rademacher 1999] addresses this limitation by providing multiple different models of an object, each associated with a single perspective, and rendering intermediate perspectives using a 3D model that is an interpolation of the models for nearby orientations. This approach, while more flexible, requires additional 3D modeling, and retains the essential 3D nature and appearance of the model.Our approach,
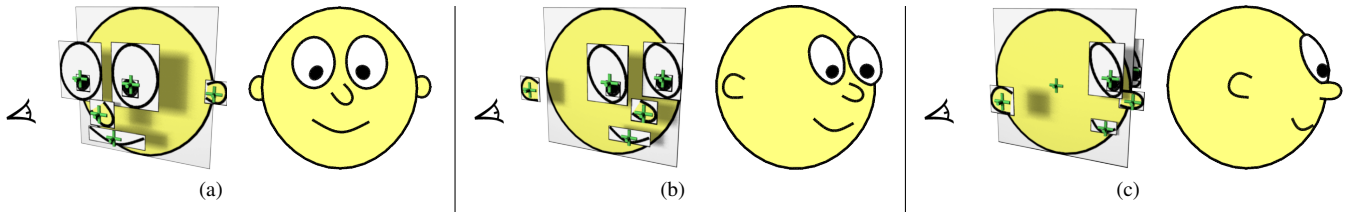
Figure 2: 2.5D interpolation: *Here we show a 2.5D model being rotated through (a) front, (b) oblique, and (c) side views. The arrangement of strokes in 3D is illustrated at the left in each view, while the 2D rendering produced is shown at right. In this cartoon, the front (a) and side (b) views were manually drawn by an artist, while the oblique view (b) was generated automatically by rotating the strokes' anchor positions (the green crosses) while interpolating the strokes' 2D shapes.*

by comparison, works entirely with 2D vector art.

Di Fiore et al. [2001] propose an approach in which a set of hand-drawn views are interpolated to generate renderings at intermediate views. In that approach, either a 3D skeleton must be manually created by the artist, or Z-ordering must be manually specified at each view. In the latter case, interpolation of shapes is simple linear interpolation. Our approach, by comparison, determines a 3D position for each stroke, and does so automatically; these 3D positions are then used both to translate the strokes along a nonlinear path and to determine their Z-ordering in novel views. Our approach also presents techniques for exploiting symmetries on the stroke level to reduce the number of views that must be manually specified, allowing the artist to easily construct models which can be fully rotated about all axes.

Our method involves a combination of translating strokes' positions based on inferred 3D positions while interpolating the strokes' shapes in 2D. Several methods have been proposed for the interpolation of 2D shapes: Sederberg and Greenwood [1992] solved the problem of vertex correspondence between two shapes by using a physically-based approach, in which strokes were likened to lengths of wire and a dynamic programming algorithm was used to minimize the amount of bending and stretching the wire undergoes in blending between two shapes. Alexa et al. [2000] tackled the problem of vertex paths, by formulating as-rigid-as-possible shape interpolation, in which paths for vertices are found that keep the volume of the interior of the stroke at intermediate points as rigid as possible. Igarashi et al. [2005b] further introduced an as-rigid-as-possible approach to interactive shape manipulation.

While these interpolation methods are purely 2D, Setz and Dyer [1996] describe View Morphing, a form of 2D image interpolation that can mimic 3D rotations between two views. However, View Morphing's approach involves warping images where corresponding 2D points represent the same 3D point. In the case of cartoons, strokes often represent silhouettes of a shape, and corresponding 2D points therefore often do not represent a single 3D point.

In our approach, we are interested in mapping vector art drawings to different points in a 2D parameterization of the orientation space, and generating plausible interpolations across that orientation space. In related work, Igarashi et al. [2005a] proposed mapping 2D vector art keyframes to positions in 3D, and allowing a user to explore the space of possible interpolations by manipulating a 3D point. The nearest keyframes to the 3D point are interpolated and the result is presented to the user. Ngo et al. [2000] investigate limiting the configuration space of interpolations between drawings to the subset that generates plausible images. They present a way for the user to manipulate the image to explore that subset of the state space without explicit knowledge of how the state space is laid out. Hsu and Lee [1994] introduced a new type of stroke for vector art which, amongst other things, could map different stroke lengths to different 2D drawings, which would be interpolated for intermediate lengths.

Several techniques aim to blend the distinction between 2D and 3D drawings. Bourguignon et al. [2001] proposed an approach in which 2D strokes are drawn onto a plane in a 3D scene. These strokes are interpreted as indicating silhouettes or contours of 3D objects, and correspondingly deform and fade into and out of visibility as the camera rotates around the scene. Yonezawa et al. [2004] present a similar approach in which 2D strokes are again embedded in 3D space, but in this case they represent silhouettes of a shape and are explicitly associated with one another, so as the camera rotates, just the appropriate silhouette can be displayed. Our approach, by comparison, combines the defined silhouettes of a stroke from multiple views using translation and interpolation to generate a full silhouette in any view.

To generate a new view of a cartoon one must not only determine strokes' shapes, but also their Z-ordering. Recently, several techniques have been proposed that seek to combine 2D drawings with depth information. Sykora et al. [2010] present a new interface to determine Z-ordering of the strokes in a cartoon using sparse depth inequalities. McCann and Pollard [2009] demonstrate how elements in a flat image can be layered with local layering constraints, rather than global ones, effectively allowing depths that vary across the object. Reversing the challenge, Eisemann et al. [2009] show how to automatically compose a 2D layered image from an 3D scene. In our approach, we determine a 3D position for each stroke, and use that to determine Z-ordering automatically (though the suggested Z-ordering can be overridden by the artist if desired).

Finally, a variety of sketch-based modeling approaches have been proposed, which aim to generate explicit 3D models from 2D drawings (e.g., [Zeleznik et al. 1996; Igarashi et al. 1999; Nealen et al. 2007; Gingold et al. 2009]). Our approach also uses 2D input to generate models which can be rotated in 3D. However, instead of generating a 3D polygonal mesh, we generate a 2.5D cartoon, which can have inconsistent appearances in different views and naturally supports a stylized drawing style.

3 Algorithm Overview

The goal of this paper is to produce a cartoon that can be rotated and rendered in any view. We take as input just 2D drawings of a cartoon in different views. We call the views which an artist has manually specified "key views" (similar to keyframes in timeline animation). The central challenge in our approach is to use these key views to determine a reasonable appearance for the cartoon in a novel view.

A naive approach would be to do simple 2D shape interpolation across the key views for each stroke. However, simple 2D inter-
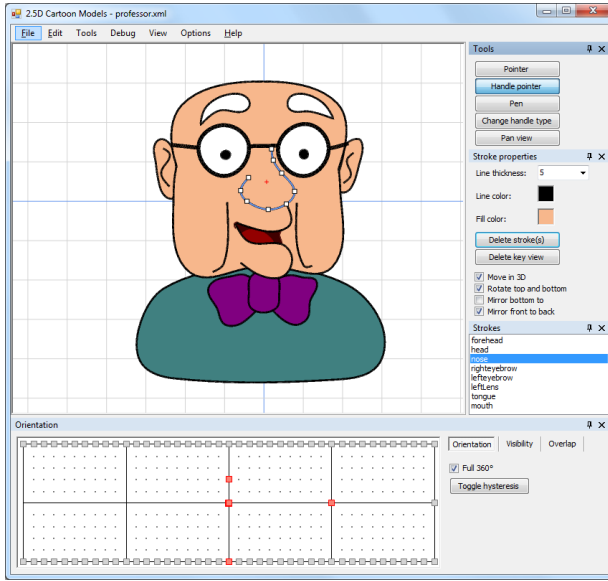
Figure 3: User interface: *The main window resembles a traditional vector-art editing interface, while the view angle control, in the lower left, parameterizes the orientation space and allows the user to select a new view to edit. The view angle control also shows existing key views for the selected stroke as red points.*

Figure 4: Parameterized orientation space: *We parameterize the space of possible views into yaw and pitch, ignoring tilt as it can be accounted for with a screen-space rotation. In this cartoon, only the views outlined in blue needed to be manually drawn by the user. Green views were automatically created by mirroring individual strokes, while red views were automatically created by rotating existing views.*

polation is unable to capture the complex nature of the paths that strokes follows in a 3D rotation. These paths curve, and strokes proceed along them at varying speed. The Z-ordering of strokes also changes during rotation, as some strokes rotate to the front and others rotate to the back. Simple 2D interpolation ignores these effects, and is unable to produce convincing results for novel views.

Instead, we investigate the challenge of *2.5D interpolation*: interpolating 2D drawings while respecting the implicit 3D structure underlying the drawings to generate an interpolated view that resembles a rotation to an intermediate viewpoint.

In determining the appearance of a stroke in a novel view, three properties must be determined: the stroke's shape, position, and Z-ordering. The core realization of our approach is that these challenges can be separated, and tackled with different tools. A stroke's shape changes in complex ways when viewed as the shifting contour of a 3D object, but can be approximated well by simple 2D interpolation. Meanwhile, strokes' positions and Z-ordering are essentially 3D properties, and while they change in complex ways when viewed in 2D, they can be easily modeled by the motion of a single 3D point.

Therefore, we propose a hybrid structure: the 2.5D cartoon model, which consists of 2D vector art strokes, each associated with a 3D position, which we call the stroke's anchor position. In general, this structure can be conceptualized as a collection of billboards positioned in 3D space, with each billboard containing a single stroke of the cartoon. To simulate a rotation between known key views, the billboards' positions are rotated in 3D about the origin, while the vector art on the billboards is interpolated with simple 2D interpolation. Thus a stroke's shape is determined by 2D interpolation across its key views, while its position and Z-ordering are determined by its associated 3D anchor position. We show a rendering of an exploded view of a 2.5D cartoon model in Figure 2.

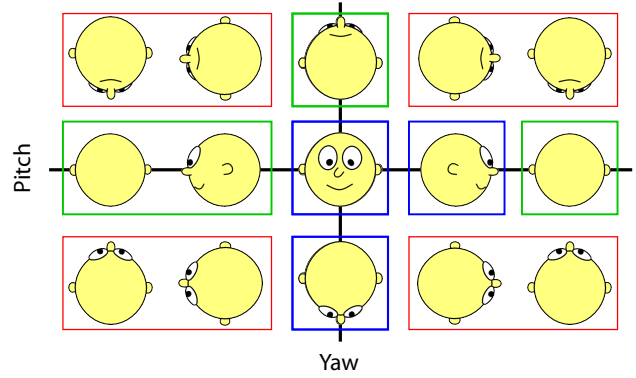In the next section, we will describe the user interface we use to construct a 2.5D cartoon model. We will then describe the details of how 2.5D interpolation is achieved, and show how we can use just three or four key views of a cartoon to generate a 2.5D cartoon model that can be rotated to any view.

4 User Interface

Our user interface, shown in Figure 3, resembles a traditional vector-art program. A user draws a cartoon using independent strokes. Unlike a traditional vector-art program, the user can at any time rotate the cartoon to edit a different view. When the user selects a new view, the interface displays its best guess as what the cartoon looks like in that view. The user can then select and redraw some or all of the strokes of the cartoon to specify their appearance in that view, creating new key views for the strokes.

Typically, an artist starts by drawing a cartoon in the front view, and then selects and redraws the strokes in the side and top views. Because a stroke must be selected before a new view for the stroke can be defined, the association between a stroke's appearances in different views is explicit.

Because we leverage redundancies and symmetries across different views of a stroke, just the top, front, and side views are usually sufficient to generate plausible appearances for the cartoon in every view. Once these views have been drawn, the artist typically spends his or her time "browsing around" the orientation space, adjusting strokes that look wrong in certain views. Unsatisfactory strokes may be adjusted either by changing their existing key views, or by redrawing them in a view in which they look wrong, adding a new key view. Additionally, the artist may adjust the visibility or Z-ordering of some strokes across different ranges of views; this can be achieved using tools we describe below.

The artist can also easily animate a 2.5D cartoon model using a provided animation timeline. The artist can enter the desired number of frames, select any frame, and redraw some or all of the strokes in the model. The result is that each stroke accumulates {time, orientation, shape} triples, which we can use to generate an animation that can be rotated around in 3D.

Please see our accompanying video for a demonstration of our editing interface. In general, we have found that most users find it quite easy to use our interface to create a 2.5D cartoon model. In terms of

time required, creating a 2.5D model is more involved than simply creating three 2D drawings, but is much less time-consuming than creating a 3D model.

We will now describe some of the tools in our interface in more detail.

4.1 Selecting a View

The artist controls the current view using the view angle control, shown in the lower-left of Figure 3. The view angle control is simply a 2D grid which parameterizes the angle space, discretized into a lattice of clickable points. The viewpoints are discretized to make it easier to return to previously specified key views, which are highlighted in red. The angle space is parameterized such that the horizontal axis of the angle control grid corresponds to yaw (rotation about the Y-axis), while the vertical axis corresponds to pitch (rotation about the X-axis). See Figure 4 for an illustration of the parameterized angle space. We have found that users are able to grasp this mapping quickly once shown the locations of the front, side, and top views.

4.2 Controlling Z-Ordering

In a traditional vector art interface, the Z-ordering of strokes can be controlled by manually moving strokes forward and backward in a global ordering. In a 2.5D model, no such global ordering exists, as the Z-ordering changes depending on the viewing angle. In our approach, once two or more views have been drawn for a stroke, we are able to automatically calculate a 3D anchor position for the stroke, and determine strokes' Z-ordering based on their anchor positions. However, the artist may wish to override this suggested Z-ordering in order to maintain high-quality renderings for all views. To allow this, we provide the overlap tool.

The overlap tool allows the artist to specify a constraint on the relative Z-ordering of two strokes over a given range of views. The artist places such a constraint by selecting two strokes, then clicking "Add > overlap region" or "Add < overlap region", and finally drawing a polygon onto the view angle control. The polygon defines the range of views for which the first seleced stroke must either be above or below the second stroke, depending on which button was clicked.

The artist may add arbitrarily many of these overlap constraints, and can therefore completely control the Z-ordering in any view. For example, if Z ordering A > B > C > D is desired in a given view, the artist can enforce that A > B, B > C, and C > D using three separate overlap constraints that include that view. In practice, the Z-ordering based on strokes' anchor positions is generally correct for most strokes; usually only a small number of manually-specified overlap constraints are necessary.

4.3 Visibility

We also provide a simple tool to allow the artist to manually specify a stroke's visibility for ranges of views. The visibility tool, similar to the overlap tool, allows the artist to draw a polygon directly onto the view angle control to specify a range of views. The artist may then specify that a particular stroke must be either visible or invisible for that range of views. Strokes are visible everywhere by default. With this tool, an artist can also create complementary groups of strokes that represent the same part of the cartoon in different stylizations, such that only one group (one appearance) of the part is visible in any one view. This may be useful for designing heavily stylized shapes that must revert to simpler appearances when viewed from oblique angles, as with the mouth in Figure 1.

4.4 Boolean Operations

Finally, an artist can also combine strokes using Boolean expressions. For example, to make the tongue stroke lie "within" the mouth stroke, the artist may first name each stroke, then select the tongue and set its Boolean expression to be "tongue ∩ mouth", which results in the tongue stroke displaying the intersection of the two strokes. Boolean operations are also useful in approximating highly concave shapes by combining multiple simple shapes into a single more complex one, as will be described in Section 6.

5 2.5D Interpolation

We will now describe in more detail how the 2.5D interpolation described in Section 3 is achieved. We will also introduce an extension which greatly reduces the number of key views that must be manually specified by an artist.

5.1 Shape Interpolation

A stroke's shape (as independent from its position and Z-ordering) in a novel view is determined by interpolating the nearest key views. We do this by first placing all key views into a 2D parameterization of the angle space, as in Figure 4. This gives us a set of 2D points representing the defined key views. We then construct a Delaunay triangulation of these points. A stroke's shape in a new view is determined by performing 2D shape interpolation across the vertices of the Delaunay triangle which the parameterization of the view lies inside. If there are no key points surrounding the given parameterized orientation, we find the nearest point on the nearest triangle edge and compute the shape at that orientation, although with the aid of derived key views (described below) this rarely occurs.

5.2 Anchor Positions

Each stroke in the 2.5D model is anchored to a single 3D anchor position which determines the stroke's position and Z-ordering. Where a stroke's anchor point should lie can be determined by considering the stroke's position in its key views. Each key view determines a 3D line passing through the center of the stroke in the direction of the camera in that view, on which the stroke's 3D position ought to lie. While these lines typically do not intersect, the ideal anchor point is one that minimizes the distance to each of these lines. We use a simple algorithm to approximate this: we start by initializing the anchor point to the origin, and iteratively move it to the average of the point's projections onto all of the 3D lines defined by the stroke's key views. We repeat this until the anchor point converges to a stable position.

Occasionally, a stroke's position will change across different views in a way that does not correspond to any real 3D position. For example, Bugs Bunny's ears always face the camera from any view. With such shapes, trying to match a 3D position to a stroke can be counter-productive. We therefore allow each stroke to optionally have no associated anchor position, in which case it is rendered based on simple 2D interpolation across its key views.

Grouping: Sometimes, several strokes in a cartoon together specify a single part of the object which must stay together as the camera rotates (e.g., the pupil and eye). If each stroke has its own anchor point, strokes which are drawn close together in every key view may nonetheless drift apart between them. To address this, we allow the user to join a selection of strokes into a group. A set of grouped strokes move relative to a single 3D anchor point, which is simply the average of the anchor points of each of the component strokes.

5.3 Derived Key Views

We can leverage symmetries to reduce the number of key views that must be manually drawn by the artist. To illustrate, consider the silhouettes of any object as viewed from the front and the back: the two silhouettes are simply horizontal mirror images of each other. Therefore, we can take a stroke's appearance in the front view and flip it to obtain its appearance in the back view. In general, we can take any artist-specified key view for a stroke and flip it to generate a derived key view for the reverse orientation. We insert these derived key views into the set of key views automatically. This tactic significantly reduces the number of key views that must be manually drawn by the artist. We show this type of derived key view in green in Figure 4.

Note that derived key views for reverse orientations can be used even if the cartoon as a whole is not a mirror image in a reversed orientation, as this is typically caused by different Z-orderings in the two views, and the Z-ordering of strokes is computed independently of their shape. However, the artist may wish to turn off this type of derived key views for strokes which have inconsistent appearances across different views, such as the mouth in Figure 1.

We also note that there exist redundancies in the way we parameterize the orientation space: all views for which $pitch = \pm\pi/2$ are in fact identical views, just rotated in image space. Therefore, when the artist specifies any key view for which $pitch = \pm\pi/2$, we automatically obtain derived key views for all other points in the view angle grid for the same $pitch$ value by rotating the given view. This effect could also be achieved by parameterizing the orientation space on a sphere; we chose our current approach as it is easier to implement and allows us to display the orientation space in the interface as a simple 2D map.

Finally, if a stroke is entirely symmetric about the vertical axis, we can simply mirror views of the stroke facing right ($yaw > 0$) to obtain views of the stroke facing left ($yaw < 0$). This also works well for stylized shapes that the artist wishes to "pop" from one orientation to another; the mouth in Figure 1 is such a case.

6 Limitations

The nature of 2.5D cartoon models imposes limitations on the types of shapes that can be represented. Because cartoons are composed out of strokes that are treated mostly independently, it is difficult to show interior contour or detail lines, as these are expected to merge with shape silhouettes at some point in rotation, an operation we do not support. In addition, shapes that are sharp or highly concave may interpolate poorly at intermediate views. Overlapping strokes can also suffer from undesired popping artifacts during rotation.

Sharp and highly concave shapes are the most difficult to represent as their contours change in complex ways during rotation that cannot be approximated with linear 2D interpolation. This issue can be addressed for some shapes by decomposing the shape into two or more mostly convex shapes, and then using Boolean union operations to combine them into what appears to be one stroke. Highly concave shapes, however, cannot always be decomposed in this fashion. Unfortunately, shapes intended as textures on the surface of a shape are especially difficult to interpolate, as they are essentially very thin concave shells. Hair is a difficult case for our system as it generally falls over the head in a way that forms a concave shell. Highly concave shapes also suffer from the limitation that we do not support partial occlusion, as with a piece of clothing wrapped around a body. These limitations can only be ameliorated by drawing in more key views.

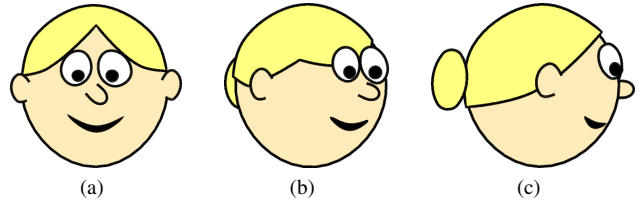Popping artifacts arise when the artist draws two shapes that over-

Figure 5: Failure case: *Although the hair stroke's two key views (a) and (c) are reasonable, the interpolated view (b) generates an implausible appearance. This is due to the shape's outline not being well approximated by linear 2D interpolation. In addition, the hair in reality undergoes partial occlusion by the head, which we do not model.*

lap at the moment their relative Z-ordering changes. While a certain amount of popping is actually expected in 2D cartoons, unintended popping artifacts can be distracting. However, only shapes that interpenetrate (in a 3D conception of the shapes) should have outlines that overlap at the moment that their relative Z-ordering changes. Popping can therefore be reduced by redrawing the strokes as noninterpenetrating shapes, so they do not overlap, or overlap as little as possible, in the views where Z-ordering changes. We wish to investigate automatic solutions to this issue in future work.

In Figure 5 we show a failure case which one stroke, the hair, interpolates poorly. Although both key views of the hair stroke are reasonable, an implausible shape is generated at an intermediate view. This is both because the shape is highly concave, and because it is partially occluded behind the head. In reality, as the head rotates, part of the hair should disappear behind the head, while other parts rotate into view. This could potentially be addressed with a more complex algorithm for shape interpolation and a model of Z-ordering that supported partial occlusion. To model this shape in our current implementation would require either a high number of key views or a decomposition into many sub-parts.

7 Results

In Figure 6, we show a number of cartoons created with our approach, showing top, front, and side key views, and one novel interpolated view. The great majority of strokes in these models were specified in either just those three views or those three plus one additional view. We give statistics in Table 1. Nonetheless, this was enough information to allow the cartoons to be rendered in any view. Rendering itself, due to the simple nature of the calculations, is far faster than real time. Please refer to our accompanying video to see these models being rotated in 3D, as well as an example of an animated 2.5D model.

8 Conclusion

We presented a new approach to non-photorealistic rendering in which 2D vector art drawings of a cartoon in different views are used to construct a 2.5D cartoon model which can render the cartoon in any angle. We described additional tools that allow the artist to fine-tune the Z-ordering and visibility of strokes in different views. We presented several models in our results section that confirm that this approach can be used to render cartoons in a way that appears hand-drawn, even from novel views.

As can be seen in Figure 6 and in our accompanying video, 2.5D cartoon models are capable of rendering organic, stylized shapes such as faces, which can be difficult to render nonphotorealistically. 2.5D models also support a wide range of styl-
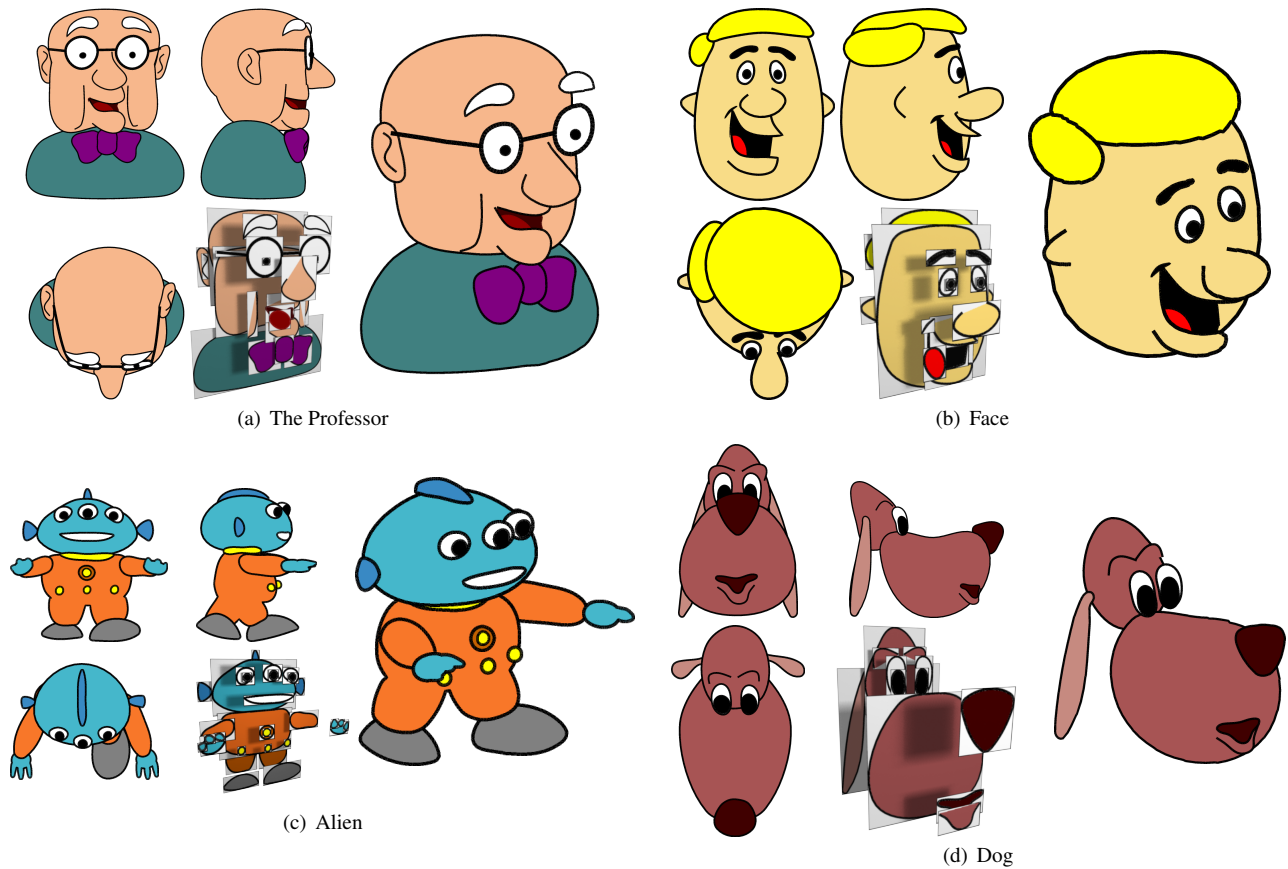
(a) The Professor

(b) Face

(c) Alien

(d) Dog

Figure 6: Results: *For each 2.5D model, we show three key views , a rendering of the 3D structure of the 2.5D model, and an interpolated view. Most strokes were drawn in just these three views or these three plus one additional view – see Table 1 for statistics.*

Model	# Strokes	Avg. # Key Views per Stroke	# Overlap Constraints
Professor	28	3.8	17
Face	18	3.6	18
Alien	32	3.125	9
Dog	14	3.2	5

Table 1: *Model statistics.*

ized 2D drawing effects that do not reflect any real 3D shape, and therefore cannot be modeled directly with 3D mesh-based approaches, but which nonetheless play an important role in the language of 2D cartooning.

Animation, which is difficult in 3D, is much simpler in 2D. Facial animation, for example, has a well-developed set of styles and methods in the realm of 2D cartooning, which are difficult to translate to a 3D model. With 2.5D cartoons, these methods and styles can be used naturally, as an animator's desired appearance for a cartoon in a given frame can be drawn in directly, without having to consider what 3D shape would produce the same effect.

We believe that this type of cartoon rendering has a wide range of applications, including in interactive entertainment, animation, and rendering on non-3D-accelerated platforms such as cell phones and Flash applications. The ease of drawing in 2D also makes it suitable for use by non-professional users.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, 157–164.

BLAIR, P. 1994. *Cartoon Animation*. Walter Foster.

BOURGUIGNON, D., CANI, M.-P., AND DRETTAKIS, G. 2001. Drawing for Illustration and Annotation in 3D. *Computer Graphics Forum 20*, 3 (September), 114–123.

DECAUDIN, P., 1996. Cartoon Looking Rendering of 3D Scenes.

EISEMANN, E., PARIS, S., AND DURAND, F. 2009. A visibility algorithm for converting 3D meshes into editable 2D vector graphics. *ACM Transactions on Graphics (TOG) 28*, 3.

FIORE, F. D., SCHAEKEN, P., ELENS, K., AND REETH, F. V., 2001. Automatic In-betweening in Computer Assisted Animation by Exploiting 2.5D Modelling Techniques.

GINGOLD, Y., IGARASHI, T., AND ZORIN, D. 2009. Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics (TOG) 28*, 5, –18.

GOOCH, B., AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. AK Peters Ltd.

HSU, S. C., AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. *International Conference on Computer Graphics and Interactive Techniques*.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A Sketching Interface for 3D Freeform Design. *International Conference on Computer Graphics and Interactive Techniques*.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. Spatial keyframing for performance-driven animation. *Symposium on Computer Animation*.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (TOG) 24*, 3.

JOHNSTON, O., AND THOMAS, F. 1981. *The Illusion of Life*. Abbeville Press.

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. 2002. WYSIWYG NPR: drawing strokes directly on 3D models. *ACM Transactions on Graphics 21*, 3 (July).

KOWALSKI, M. A., MARKOSIAN, L., NORTHRUP, J. D., BOURDEV, L., BARZEL, R., HOLDEN, L. S., AND HUGHES, J. F. 1999. Art-based rendering of fur, grass, and trees. *International Conference on Computer Graphics and Interactive Techniques*.

MCCANN, J., AND POLLARD, N. 2009. Local layering. *International Conference on Computer Graphics and Interactive Techniques 28*, 3.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh:designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (TOG) 26*, 3.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. *International Conference on Computer Graphics and Interactive Techniques*.

NORTHRUP, J. D., AND MARKOSIAN, L. 2000. Artistic silhouettes:a hybrid approach. *Non-Photorealistic Animation and Rendering*.

RADEMACHER, P. 1999. View-dependent geometry. *International Conference on Computer Graphics and Interactive Techniques*.

SEDERBERG, T. W., AND GREENWOOD, E. 1992. A physically based approach to 2D shape blending. *International Conference on Computer Graphics and Interactive Techniques*.

SEITZ, S. M., AND DYER, C. R. 1996. View morphing. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, 21–30.

SÝKORA, D., SEDLÁČEK, D., JINCHAO, S., DINGLIANA, J., AND COLLINS, S. 2010. Adding Depth to Cartoons Using Sparse Depth (In)equalities. *Computer Graphics Forum 29*, 2.

YONEZAWA, K., TAKAHASHI, S., AND SHIBAYAMA, E. 2004. SilF: a sketching tool for cartoon-like pseudo-3D illustrations based on 3D outlines. *International Conference on Computer Graphics and Interactive Techniques*.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. SKETCH: An Interface for Sketching 3D Scenes. *International Conference on Computer Graphics and Interactive Techniques*.