

DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling

XIAO GUANG HAN, CHANG GAO, and YIZHOU YU, The University of Hong Kong

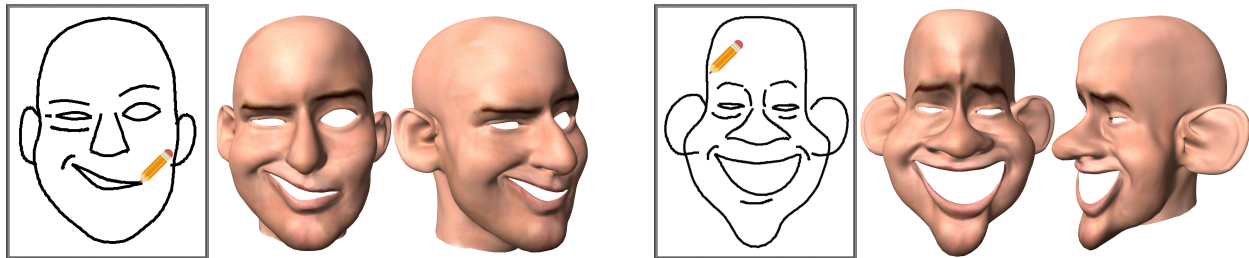


Fig. 1. Using our sketching system, an amateur user can create 3D face or caricature models with complicated shape and expression in a few minutes. Both models shown here were created in less than 10 minutes by a user without any prior drawing and modeling experiences.

Face modeling has been paid much attention in the field of visual computing. There exist many scenarios, including cartoon characters, avatars for social media, 3D face caricatures as well as face-related art and design, where low-cost interactive face modeling is a popular approach especially among amateur users. In this paper, we propose a deep learning based sketching system for 3D face and caricature modeling. This system has a labor-efficient sketching interface, that allows the user to draw freehand imprecise yet expressive 2D lines representing the contours of facial features. A novel CNN based deep regression network is designed for inferring 3D face models from 2D sketches. Our network fuses both CNN and shape based features of the input sketch, and has two independent branches of fully connected layers generating independent subsets of coefficients for a bilinear face representation. Our system also supports gesture based interactions for users to further manipulate initial face models. Both user studies and numerical results indicate that our sketching system can help users create face models quickly and effectively. A significantly expanded face database with diverse identities, expressions and levels of exaggeration is constructed to promote further research and evaluation of face modeling techniques.

CCS Concepts: • **Computing methodologies** → **Graphics systems and interfaces**; **Shape modeling**;

Additional Key Words and Phrases: Face Modeling, Face Database, Deep Learning, Face Caricatures, Gestures, Sketch-Based Modeling

ACM Reference format:

Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Trans. Graph.* 36, 4, Article 126 (July 2017), 12 pages. <https://doi.org/10.1145/3072959.3073629>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 Association for Computing Machinery.

0730-0301/2017/7-ART 126 \$15.00
<https://doi.org/10.1145/3072959.3073629>

1 INTRODUCTION

Face modeling has been a popular research topic in the field of visual computing, and different approaches, including real face digitization and interactive face modeling, have been attempted in response to a variety of application scenarios. While high-end applications, such as virtual characters in feature films, demand high-fidelity face models acquired from the real world, there exist many scenarios, including cartoon characters, custom-made avatars for games and social media, 3D face caricatures as well as face-related art and design, where low-cost interactive modeling is still a mainstream approach especially among amateur users. Nevertheless, faces have rich geometric variations due to diverse identities and expressions, and interactively creating a decent 3D face model is a labor-intensive and time-consuming task even for a skilled artist with the help of well-developed software such as MAYA or ZBrush. Thus creating expressive face models with a minimal amount of labor is still a major challenge in interactive face modeling.

Let us leave 3D faces for a moment and think about how we draw a 2D face. It is a natural choice for us to quickly draw the outermost silhouette first and then the contours of various facial features including eyes, nose and mouth. Such a set of silhouettes and contours already give a very good depiction of the underlying face even though they are merely a sparse set of lines. Note that such silhouettes and contours also exist on a 3D face except that they have extra depth information and the silhouette is view-dependent. However, even when the 3D silhouette and contours are fully specified, the shape of the facial regions in-between these sparse lines are still unknown. Fortunately, the 3D shape of these regions might be highly correlated with the 2D and 3D shape of the silhouette and contours. For example, there exist strong correlations between the nose contour and the overall shape of the nose, and also between the mouth contour and the shape of the cheeks. Therefore, it is crucial to learn such correlations from data and let them guide labor-efficient face modeling.

Example-Based Expressive Animation of 2D Rigid Bodies

MAREK DVOROŽŇÁK, Czech Technical University in Prague, Faculty of Electrical Engineering

PIERRE BÉNARD, LaBRI (UMR 5800, CNRS, Univ. Bordeaux), France

PASCAL BARLA, Inria Bordeaux Sud-Ouest, France

OLIVER WANG, Adobe Research

DANIEL SÝKORA, Czech Technical University in Prague, Faculty of Electrical Engineering

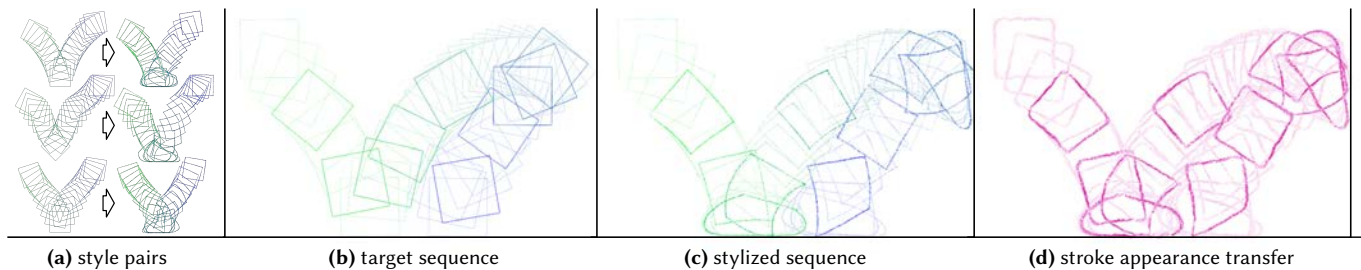


Fig. 1. Given a small set of exemplars consisting of computer-generated and hand-drawn 2D animation pairs (a), our method transfers to a new target sequence produced by physical simulation (b) both the high-level deformations and fine-scale appearance variations (c) present in the example animations. Optionally, the final appearance of the drawings can be modified by re-synthesizing different stroke textures (d).

We present a novel approach to facilitate the creation of stylized 2D rigid body animations. Our approach can handle multiple rigid objects following complex physically-simulated trajectories with collisions, while retaining a unique artistic style directly specified by the user. Starting with an existing target animation (e.g., produced by a physical simulation engine) an artist interactively draws over a sparse set of frames, and the desired appearance and motion stylization is automatically propagated to the rest of the sequence. The stylization process may also be performed in an off-line batch process from a small set of drawn sequences. To achieve these goals, we combine parametric deformation synthesis that generalizes and reuses hand-drawn exemplars, with non-parametric techniques that enhance the hand-drawn appearance of the synthesized sequence. We demonstrate the potential of our method on various complex rigid body animations which are created with an expressive hand-drawn look using notably less manual interventions as compared to traditional techniques.

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

Additional Key Words and Phrases: 2D animation, example-based synthesis

ACM Reference format:

Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sýkora. 2017. Example-Based Expressive Animation of 2D Rigid Bodies. *ACM Trans. Graph.* 36, 4, Article 127 (July 2017), 10 pages.

DOI: <http://dx.doi.org/10.1145/3072959.3073611>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 ACM. 0730-0301/2017/7-ART127 \$15.00
DOI: <http://dx.doi.org/10.1145/3072959.3073611>

1 INTRODUCTION

Despite the recent success of computer-generated animations, traditional hand-drawn approaches often yield more expressive and stylized looks than those produced with the currently available digital tools. However, creating hand-drawn animations is a tedious process that requires years of training by an artist and countless hours of labor. Furthermore, style is a highly personalized concept, and two different artists never animate exactly in the same way. As a result, example-based stylization has been a long-standing goal in computer graphics.

In this work we focus on rigid bodies, which are particularly challenging to animate by hand, since multiple objects may collide and rebound in ways that are difficult to plan in advance. Conversely, using physical simulation, computer-based methods can quickly give rise to rigid body animations with realistic trajectories, but ones that lack *expressiveness*. Our main goal is therefore to combine the ease of use of computer-simulated 2D rigid body animations with the expressive qualities of hand-drawn techniques.

To accomplish this goal, we have a number of added requirements. First, editability is of paramount importance to animators, and an ideal solution should work iteratively, always providing the artist with the ability to refine the current solution. Second, producing each hand-drawn frame is time consuming, so a practical example-based 2D animation system should be able to generalize from a very limited set of artistic inputs, while being able to apply these edits seamlessly into the dense set of final target frames. These two requirements make example-based 2D animation out of reach of current data-driven machine learning techniques, due to the scarcity of data (tens of exemplars, rather than tens of thousands), and uniqueness of each style.

Instead, our approach is inspired by a workflow that is widespread among both traditional and digital animators. A 2D animation is

Skippy: Single View 3D Curve Interactive Modeling

VOJTĚCH KRS, Purdue University
ERSIN YUMER, Adobe Research
NATHAN CARR, Adobe Research
BEDRICH BENES, Purdue University
RADOMÍR MĚCH, Adobe Research

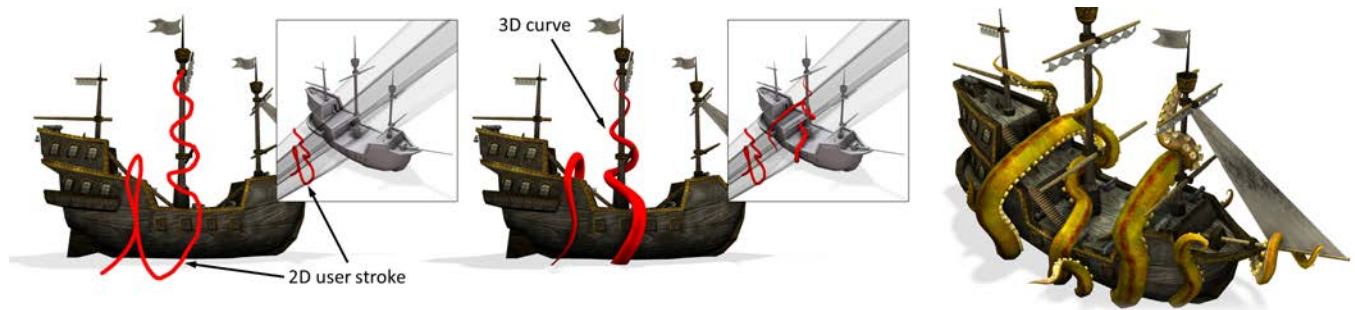


Fig. 1. The user draws a 2D stroke in front of the 3D model (left). The 2D stroke is converted into a 3D curve (middle). A complete 3D model from multiple curves is generated within a few seconds (right).

We introduce Skippy, a novel algorithm for 3D interactive curve modeling from a single view. While positioning curves in space can be a tedious task, our rapid sketching algorithm allows users to draw curves in and around existing geometry in a controllable manner. The key insight behind our system is to automatically infer the 3D curve coordinates by enumerating a large set of potential curve trajectories. More specifically, we partition 2D strokes into continuous segments that land both *on* and *off* the geometry, duplicating segments that could be placed in front or behind, to form a directed graph. We use distance fields to estimate 3D coordinates for our curve segments and solve for an optimally smooth path that follows the curvature of the scene geometry while avoiding intersections. Using our curve design framework we present a collection of novel editing operations allowing artists to rapidly explore and refine the combinatorial space of solutions. Furthermore, we include the quick placement of transient geometry to aid in guiding the 3D curve. Finally we demonstrate our interactive design curve system on a variety of applications including geometric modeling, and camera motion path planning.

CCS Concepts: • **Computing methodologies** → **Parametric curve and surface models**; • **Human-centered computing** → **Graphical user interfaces**;

This work has been sponsored by Adobe Research and by the National Science Foundation grant #1606396 *Haptic-Based Learning Experiences as Cognitive Mediators for Conceptual Understanding and Representational Competence in Engineering Education* Author's addresses: Vojtěch Krs and Bedrich Benes, Computer Graphics Technology, 401 N Grant St, West Lafayette, IN 47907. Ersin Yumer, Nathan Carr and Radomír Měch, Adobe Research, 345 Park Ave, San Jose, CA 95110.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 ACM. 0730-0301/2017/7-ART128 \$15.00
DOI: <http://dx.doi.org/10.1145/3072959.3073603>

Additional Key Words and Phrases: Single View, 3D Curve, Geometric Modeling

ACM Reference format:

Vojtěch Krs, Ersin Yumer, Nathan Carr, Bedrich Benes, and Radomír Měch. 2017. Skippy: Single View 3D Curve Interactive Modeling. *ACM Trans. Graph.* 36, 4, Article 128 (July 2017), 12 pages.
DOI: <http://dx.doi.org/10.1145/3072959.3073603>

1 INTRODUCTION

Computer graphics has achieved impressive results in areas such as rendering and computer animation. However, 3D modeling still poses many challenges; one of them is expressing user intent by simple means. User interaction, which is at the heart of modeling, is where most of the related complexity still exists. This is mainly due to the fact that most input and display devices currently in use are 2D, which is not intuitive to human interaction with the world, where we operate and think in 3D. To overcome the loss of depth, the user is usually forced to change the viewpoint, rotate the object, or use multiple viewports at once [Bae et al. 2008]; which can lead to a loss of efficiency.

A particularly difficult problem is drawing of 3D curves. These *space curves* are important to a variety of tasks such as planning of trajectories of dynamic objects, for example, particle systems or virtual cameras, design of curved surface patches, such as NURBS, or swept surfaces, or generalized cylinders. The main problem of drawing 3D curves in 2D is that there is an infinite number of possible configurations of the curve in the missing dimension, and, as explored by Schmidt et al. [2009a], even expert users have trouble with drawing 3D objects and curves. The foreshortening caused by perspective projection is especially difficult to get right and the resulting objects hardly match user's intent. While the use of shadows as visual depth cues has been shown to improve spatial

κ -Curves: Interpolation at Local Maximum Curvature

ZHIPEI YAN, Texas A&M University
STEPHEN SCHILLER, Adobe Research
GREGG WILENSKY, Adobe
NATHAN CARR, Adobe Research
SCOTT SCHAEFER, Texas A&M University

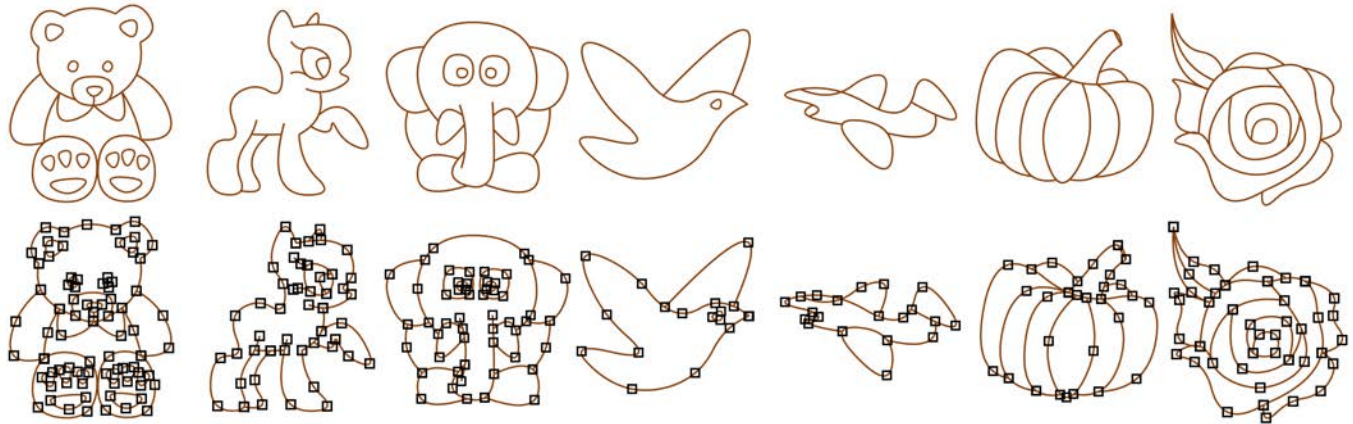


Fig. 1. Top row shows example shapes made from the control points below. In all cases, local maxima of curvature only appear at the control points, and the curves are G^2 almost everywhere.

We present a method for constructing almost-everywhere curvature-continuous, piecewise-quadratic curves that interpolate a list of control points and have local maxima of curvature only at the control points. Our premise is that salient features of the curve should occur only at control points to avoid the creation of features unintended by the artist. While many artists prefer to use interpolated control points, the creation of artifacts, such as loops and cusps, away from control points has limited the use of these types of curves. By enforcing the maximum curvature property, loops and cusps cannot be created unless the artist intends for them to be.

To create such curves, we focus on piecewise quadratic curves, which can have only one maximum curvature point. We provide a simple, iterative optimization that creates quadratic curves, one per interior control point, that meet with G^2 continuity everywhere except at inflection points of the curve where the curves are G^1 . Despite the nonlinear nature of curvature, our curves only obtain local maxima of the absolute value of curvature only at interpolated control points.

CCS Concepts: •Computing methodologies → Parametric curve and surface models;

Additional Key Words and Phrases: interpolatory curves, monotonic curvature, curvature continuity

This work was supported by NSF Career award IIS 1148976.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 ACM. 0730-0301/2017/7-ART129 \$15.00
DOI: <http://dx.doi.org/10.1145/3072959.3073692>

ACM Reference format:

Zhipei Yan, Stephen Schiller, Gregg Wilensky, Nathan Carr, and Scott Schaefer. 2017. κ -Curves: Interpolation at Local Maximum Curvature. *ACM Trans. Graph.* 36, 4, Article 129 (July 2017), 7 pages.
DOI: <http://dx.doi.org/10.1145/3072959.3073692>

1 INTRODUCTION

Curve modeling has a long history in computer graphics, finding use in drawing, sketching, data fitting, interpolation, as well as animation. This rich application space has led to decades of research for both representing and modifying curves. The goal of such curve representations is to provide the user with control over the shape of the curve while building a curve that has certain geometric properties. These properties may include smoothness, interpolation of various points, and locality.

In this paper we focus on interpolatory curves; that is, curves that interpolate their control points. While much research has concentrated on approximating curves, many users prefer direct control over salient geometric features of the curve such as the position of the curve. Yet interpolatory curves have a maligned past as they can often generate geometric features such as cusps and loops away from control points that the user has a hard time controlling (see Figure 2).

Our premise is that salient geometric features should appear only at control points for interpolatory curves. Position is one such example of a feature that is automatically enforced in interpolatory curve constructions. However, the question is then: what other features should appear only at control points? Levien et al. [Levien