Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation

MAREK DVOROŽŇÁK and DANIEL SÝKORA, Czech Technical University in Prague, Faculty of Electrical Engineering
CASSIDY CURTIS, Google Research
BRIAN CURLESS, Google Research and University of Washington
OLGA SORKINE-HORNUNG, ETH Zurich
DAVID SALESIN, Google Research

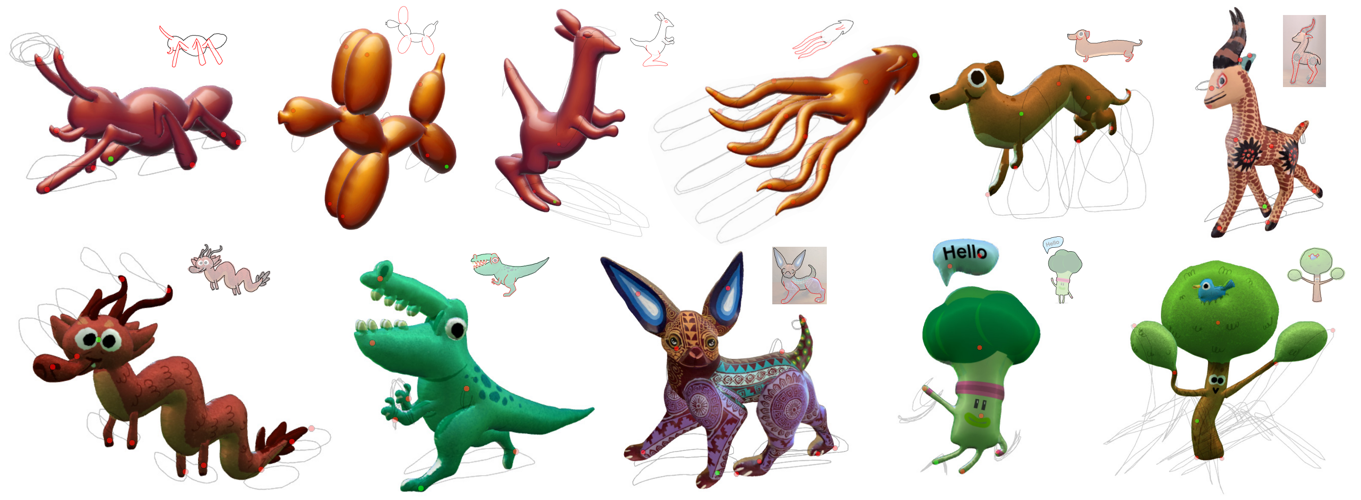

Fig. 1. A selection of animated characters created using Monster Mash by animators who participated in our informal user study. For each example, the original hand-drawn outlines are visible in the inset above (superimposed over the original drawing or a source photo). From these outlines, Monster Mash can inflate a smooth, consistent 3D model that can immediately be animated using several control points (red and green dots). Their trajectories are visualised as grey curves. See our supplementary videos for these characters in motion. Colored source drawings by Hélène Leroux and Neth Nom.

We present a new framework for sketch-based modeling and animation of 3D organic shapes that can work entirely in an intuitive 2D domain, enabling a playful, casual experience. Unlike previous sketch-based tools, our approach does not require a tedious part-based multi-view workflow with the explicit specification of an animation rig. Instead, we combine 3D inflation with a novel rigidity-preserving, layered deformation model, ARAP-L, to produce a smooth 3D mesh that is immediately ready for animation. Moreover, the resulting model can be animated from a single viewpoint — and without the need to handle unwanted inter-penetrations, as required by previous approaches. We demonstrate the benefit of our approach on a variety of examples produced by inexperienced users as well as professional animators. For less experienced users, our single-view approach offers a simpler modeling and animating experience than working in a 3D environment, while for professionals, it offers a quick and casual workspace for ideation.

CCS Concepts: • **Computing methodologies → Mesh models**; **Animation**; • **Human-centered computing → Graphical user interfaces**.

Additional Key Words and Phrases: sketch-based modelling, 2D-to-3D inflation, casual animation

ACM Reference Format:
Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation. *ACM Trans. Graph.* 39, 6, Article 214 (December 2020), 12 pages. https://doi.org/10.1145/3414685.3417805

Authors' addresses: Marek Dvorožňák, dvoromar@fel.cvut.cz; Daniel Sýkora, sykorad@fel.cvut.cz, Czech Technical University in Prague, Faculty of Electrical Engineering, Karlovo náměstí 13, Praha 2, Czech Republic, 121 35; Cassidy Curtis, cassidycurtis@google.com, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, 94043; Brian Curless, curless@google.com, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, 94043, University of Washington, Paul G. Allen Center 664, Seattle, WA, 98195; Olga Sorkine-Hornung, sorkine@inf.ethz.ch, ETH Zurich, Universitaetstrasse 6, Zurich, Switzerland, 8092; David Salesin, salesin@google.com, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, 94043.

1 INTRODUCTION

Most forms of artistic expression span a range from formal to casual: for example, poetry includes everything from the villanelle, whose strict rules demand painstaking construction, to freestyle rap, composed in the moment of speaking. Likewise, music can be a structured composition or an impromptu jam session, actors can memorize lines or perform live improvisation, and a visual artist can spend months on an oil painting, or do a 30-second gesture drawing. What the casual forms have in common is that they are fast, immediate, intuitive, and low-risk, and they allow an artist to produce a complete result starting from scratch. A casual mode of expression is crucial to the creative process: people are the most creative when they are in a state of "flow" [Csikszentmihalyi 1991], which requires being free from technical obstacles, and especially being free from the fear of making a mistake.

The art form we call animation has never truly had a casual mode. Even the simplest animation techniques are extremely time-consuming, and in 3D especially so, because there are so many intermediate steps — modeling, defining skeletal joints and deformation parameters, posing and setting keyframes — between an idea and a finished result. There are certainly no tools that allow an animator to go from a blank page to a full 3D animated character in anything like real time. Animation's lack of a native casual mode drives animators to do their creative exploration in other mediums, like thumbnail sketches, storyboards, or live-action video.

In this paper we propose a playful 3D modeling and animation tool, called "Monster Mash," that enables a casual creative experience for ideation and prototyping. The user sketches out a character, and the software automatically converts it to a soft, deformable 3D model, which the user can immediately animate, simply by grabbing parts of it and moving them around in real time.

In Monster Mash, 3D models can be created and animated from a single view, without the need to specify an animation rig, perform merging of model parts, or explicitly check consistency in depth during the animation. The modeling process is performed jointly on the entire mesh and is directly coupled with a set of dynamic depth-ordering and positional constraints that together drive our novel rigidity-preserving, layered deformation model, *ARAP-L*. Since these constraints are active during animation they constantly influence the object's shape and help to preserve consistency in depth despite the fact that the modeling as well as animation is performed from a single viewpoint. To the best of our knowledge, Monster Mash is the first sketch-based tool capable of creating and animating a smooth, consistent 3D model from a single viewpoint within seconds, thus unlocking the potential for casual creation.

2 RELATED WORK

A key research direction that represents a cornerstone for casual creation of animated 3D models is the sketch-based modeling approach pioneered by Igarashi et al. [1999] in their system called Teddy. It allows the user to specify a 3D model as a set of components whose silhouettes are drawn in 2D and then inflated into 3D. Individual parts are positioned in 3D space with the help of different viewpoints and then merged. Thanks to this intuitive process, complex 3D models can be created from a set of hand-drawn strokes with a relatively small amount of interaction. Follow-up works study various ways the individual parts can be created through inflation, manipulated, or joined together. Tai et al. [2004] use convolution surfaces for inflation, Schmidt et al. [2005] propose blob trees, and Nealen et al. [2007] allow the user to specify 3D control curves on the surface of individual components, which enable better shape manipulation and specification of contact points. Bernhardt et al. [2008] introduce a region-painting metaphor, which uses implicit surfaces without the need to perform optimization. Gingold et al. [2009] use a variety of structural annotations that better define the shape, as well as interconnections of individual components. Finally, Rivers et al. [2010] relaxes the necessity of producing the 3D geometry while still being able to view the model from different viewpoints. They decompose the input vector drawings into a set of billboards whose centroids are interpolated in 3D space while the shapes of the strokes inside the billboards undergo only a 2D morph. The resulting transition preserves the semblance of a convincing 3D motion. Despite the success of these modeling techniques, they are all still rather close to the traditional 3D CAD modeling workflow, in which careful step-by-step viewpoint planning or tedious manipulation in 3D space is required.

For casual creation, interaction in 2D space is notably more appealing, as there is no need to leave the intuitive 2D mindset and perform manipulation in 3D. Karpenko and Hughes [2006] address this requirement in their SmoothSketch system. They first analyze the input sketch to infer the shape of individual components and then estimate how they are connected in 3D space. Although the complexity of the 3D models that SmoothSketch can deliver is relatively limited, the method demonstrates that a variety of modeling tasks can be performed automatically with only 2D interactions. Other works adopt a similar paradigm. Turquin et al. [2007] propose a single-view system tailored to cloth modeling that uses function-specific strokes to design garment breakpoints and folds. Olsen and Samavati [2011] add automatic classification of input strokes while Cordier et al. [2011] leverage object symmetries.

The simplified 2D scenario opens the possibility to use bas-relief approximation [Joshi and Carr 2008] or normal map estimation [Tuan et al. 2015] which provide enough 3D structure for applications when only shading effects are added to the input sketch or when the camera undergoes a small out-of-plane rotation [Yeh et al. 2017]. Even in this simplified scenario complex layered structures can still be created [Dvorožňák et al. 2018; Sýkora et al. 2014]. These layered methods, however, require estimation of relative depth order of individual layers, their absolute positioning in depth, and computation of smooth interconnections.

In a similar vein to bas-relief modeling, Entem et al. [2015] propose a framework that exploits symmetries, depth order estimation and implicit surfaces to produce full 3D models of animals. Their approach was also extended to handle automatic decomposition of an input drawing into a set of salient parts [Entem et al. 2019]. Similarily, Ramos et al. [2018] build a half-edge structure from the input sketch to extract structural parts and their symmetries, which are then converted into implicit surfaces to inflate the model. In recent work, Bobenrieth et al. [2020] further assume that the input sketch consists of a set of solid lines defining visible silhouettes and dashed lines representing hidden outlines. Their method can then produce 3D elements and merge them together to form the final 3D shape. Although these techniques can be used for single-view modeling, they still subdivide the modeling process into a set of sub-problems treated independently — i.e., they first perform inflation of individual components, then position them in depth, and finally determine the resulting intersections or perform a merging operation to produce the final 3D mesh. As a result, the output surfaces contain visible transitions between individual parts, which

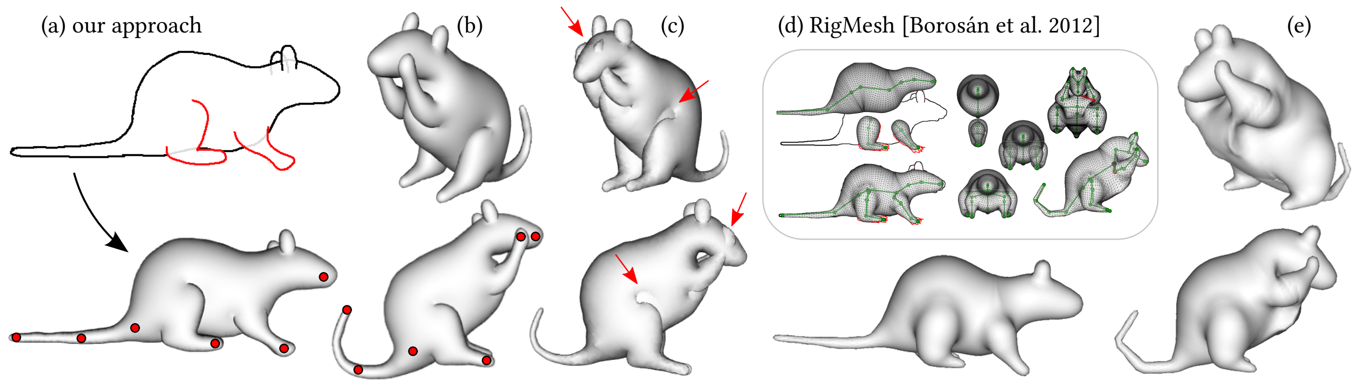

Fig. 2. Monster Mash in action: (a) From a sparse set of strokes drawn solely in a 2D plane, some of which are marked in red to indicate symmetric components, our system can inflate and directly animate a complete, consistent 3D model (b) through a set of user-specified control points (red dots). A key contribution of Monster Mash is that it introduces a new rigidity- and layer-preserving deformation model (ARAP-L) that helps to maintain consistency of the resulting 3D mesh during the modeling phase as well as animation. In contrast to using standard as-rigid-as-possible (ARAP) deformation, our deformation model avoids creating certain undesirable shape details, which may become visible when applying ARAP directly to the rest pose (c). Also, since the position of control points is specified in 2D, using ARAP alone some components of the resulting model may collide and penetrate due to depth ambiguities. All of these artifacts are automatically eliminated using ARAP-L. Our new approach can significantly lower the amount of interaction required to produce a consistent, animated 3D mesh, making the creative process fluid and unhindered. By contrast, previous sketch-based tools that allow for modeling and animation require users to work in multiple viewpoints and build an animation rig before the model can be inflated and animated (d). Moreover, they also suffer from various shape-modeling artifacts, caused by merging the individual mesh components in a post-process (e). In Monster Mash, inflation is applied jointly on the entire model, which leads to notably smoother results, as shown in (b).

look unnatural. Moreover, all these techniques consider solely the modeling phase: issues arising when the inflated models need to be animated are not addressed.

An alternative approach to single-view sketch-based modeling is presented by Xu et al. [2014] in their True2Form system, where a specific form of product design sketches is assumed as input. Such sketches contain auxiliary cross-section curves that are exploited by the algorithm to interpret the final consistent 3D shape. Similarly, in BendSketch [Li et al. 2017], in addition to silhouettes, the input drawings include special strokes to represent the bending directions of the final 3D form. Although these techniques are capable of producing quite complex 3D shapes from a single-view sketch, they require experienced users who are familiar with drawing cross-section curves or can imagine the desired bending of the final surface.

With recent progress in deep neural networks, data-driven approaches become a viable basis for sketch-based modeling. Lun et al. [2017] predict the depth and normal maps from input sketches drawn from multiple viewpoints. From these maps, a point cloud is reconstructed, which is further refined to obtain the target 3D mesh. Delanoy et al. [2018] train a 2D-to-3D network that can directly convert multiple sketches drawn from different viewpoints into a volumetric representation. Finally, Li et al. [2018] infer a depth map and a normal map from a single-view sketch with additional depth and curvature hints. Multiple viewpoints can be provided to progressively improve the resulting 3D mesh. In contrast to purely geometric approaches, neural-based techniques require training datasets, which influences the ability of the network to predict the resulting 3D shape. Moreover, if the user is not satisfied with the result, it may be challenging to gain precise control over the inference process.

A common drawback of both the purely geometric and data-driven techniques is that their final result is a 3D mesh that needs to undergo an additional rigging phase before it can be animated. This limitation is recently addressed by sketch-based modeling tools that combine model creation with preparation of an animation rig. In RigMesh [Borosán et al. 2012], ArtiSketch [Levi and Gotsman 2013] or AniMesh [Jin et al. 2015], a multi-view part-based sketching workflow is used to design a rigged 3D mesh that can be subsequently animated. A key idea of these works is to build a skeleton on the fly during the creation of individual model parts. When a specific part is finished, a bone is automatically assigned to it and subsequently connected with other nearby bones to form skeletal joints. Alternatively, Bessmeltsev et al. [2015] let the user prepare the skeleton in advance to accompany an existing 2D sketch. This auxiliary input not only facilitates subsequent animation, but also helps to reduce depth ambiguities during the modeling phase. Although the simultaneous rig creation enables direct animation after the model is finished, it still requires using multiple viewpoints and carefully planning the skeletal structure, which could be unintuitive and confusing for novice users who may not be familiar with the workflow used in professional animation tools. Even for professional users, the burden of rig preparation could discourage them from the creative process of fast ideation. Moreover, during the subsequent animation the user needs to leave the intuitive 2D domain again to check consistency of individual moving parts in 3D and avoid their unintentional collision or penetration. Effects such as body parts pressing up against each other and affecting the final shape

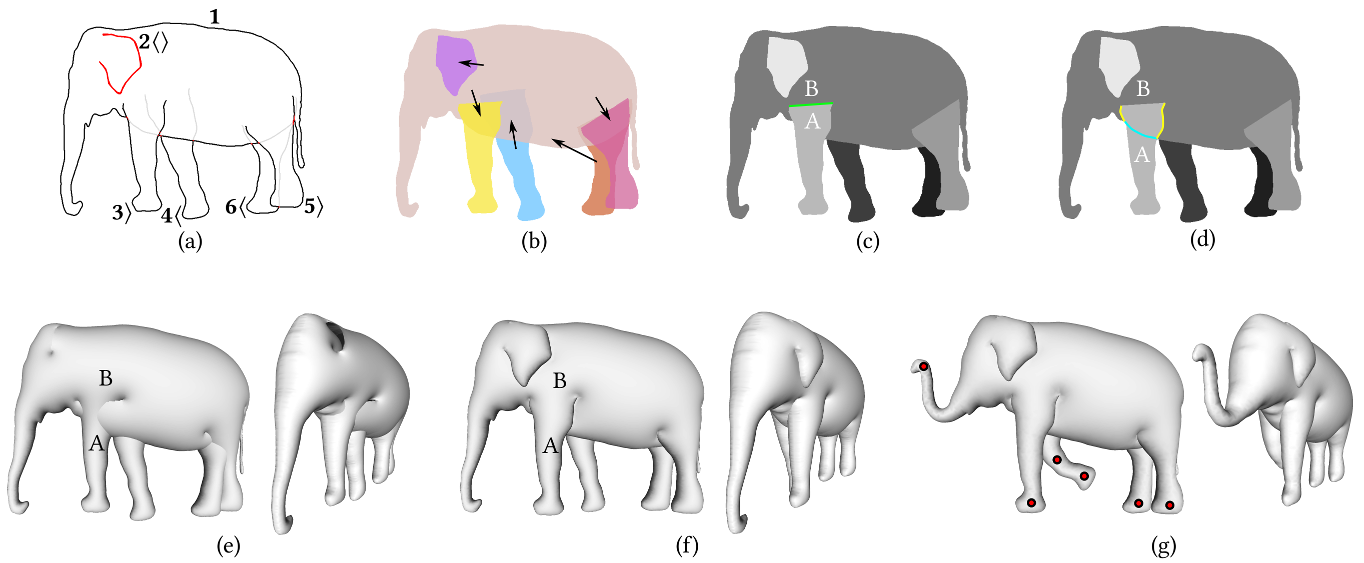

Fig. 3. Full pipeline: (a) A user draws a set of strokes representing semantically meaningful parts of the target 3D model. For each stroke, the user indicates whether the part it specifies lies in front of ("⟩"), behind ("⟨"), or is symmetric to ("⟨⟩") — that is, should be replicated on both sides of—the previously drawn parts. Open strokes are closed automatically; if all or a portion of an added closing curve for one part is on top of (or behind) another part, then that portion of the curve becomes a merging boundary between the parts. The partial depth-ordering constraints (b), indicated by arrows pointing to the closer part, are used to compute the total depth order of all the parts, represented by the brightness of the part in (c). The green line in (c) closes the boundary of part A and, as it is on top of B, is a "merging" boundary where the domains of A and B will be stitched. In (d), the yellow and cyan lines indicate curves where inequality constraints will be maintained when constructing a 3D model. Mesh points along the yellow curves belong to A, and their depths are constrained to lie in front of all points in B, while points along the cyan curve belong to B and have depths constrained to lie behind points in A. Note that after constructing and animating the model, part A can move relative to part B, and thus the yellow and cyan curves will be updated dynamically. After inflation (e), we have a 3D shape with all parts stitched together, so that, for example, part A smoothly connects to part B (joined along the green line indicated in (c)). However, the Dirichlet boundary constraints force all surface parts to be height fields "rooted" in the $z = 0$ plane, as seen in the rotated view in (e). Imposing the ordering constraints with our ARAP-L deformation results in a mesh (f) with parts correctly moved in front of (or behind) other parts. For example, the points from part A on the yellow curve in (d) are now above part B, and the points from part B on the cyan curve in (d) are now below part A. Finally, as shown in (g), the user can interactively deform the mesh into a new pose with point constraints, while automatically maintaining ordering constraints, all in the same ARAP-L framework.

are not supported, which may lead to disturbing modeling artifacts (see Fig. 2).

In our workflow, we adopt the simplicity and intuitiveness of the shape manipulation techniques based on as-rigid-as-possible (ARAP) deformation [Igarashi et al. 2005; Sorkine and Alexa 2007]. These methods present a viable alternative to traditional rig-based control while having a comparable expressive power [Jacobson et al. 2012]. Their key advantage is that they do not require a predefined skeletal structure, accurate skinning, or complex algorithms to drive direct manipulation of the skeleton (e.g., inverse kinematics). Although ARAP deformation could be applied to a 3D model inflated by a sketch-based modeling system, it may fail to produce desirable results due to the shape-preserving nature of the ARAP model: for example, an impression in the surface, caused by an attached model part, may persist even when the attached part is moved to a different position. Also, since the manipulation is performed in a 2D plane, parts could collide with each other in 3D space and produce inconsistent results. (See the comparison of the results produced by our approach versus the original ARAP deformation in Fig. 2.)

3 OUR APPROACH

Our sketch-based modeling and animation tool builds upon an established part-based metaphor [Igarashi et al. 1999] that allows a user to quickly outline the basic structure of the target 3D model. In this workflow, individual parts specify the shape of an unoccluded 2D projection of a semantically meaningful part of the target 3D model. Our method, however, differs in the way those user-specified parts are subsequently processed. In most of the previous sketch-based modeling approaches, each part is inflated independently and then merged with the rest of the 3D model. In our workflow, we extend the method of Dvorožňák et al. [2018], in which parts are merged in advance before the model construction process starts, leading to a more natural-looking shape. However, a key drawback of Dvorožňák et al.'s method is that it considers only 2.5D bas-relief rather than full 3D meshes, and also does not take any sort of animation into account. In our approach we extend the original concept into a fully 3D scenario and provide the ability to animate the mesh while dynamically updating deformations resulting from the layering of parts on top of one another.

To provide an input to our method, the user starts with an empty canvas, or (optionally) an existing drawing or photograph over which strokes can be traced. A tablet or a mouse can be used to sequentially draw a set of open and closed 2D strokes (see Fig. 3a). These strokes specify the shape of individual parts (Fig. 3b). The user can indicate (e.g., with specific hotkeys) whether those parts are positioned in front of or behind already drawn parts (see arrows in Fig. 3a). Thanks to this interface, relative depth ordering of individual parts is established (Fig. 3b). Parts can also be marked as bilaterally symmetric (e.g., the elephant's ear in Fig. 3a). Those parts are automatically duplicated, and their relative depth order is changed so that the symmetric replica lies on the opposite side of the target 3D object. If the drawn stroke is open (e.g., the elephant's front leg in Fig. 3a), it is automatically closed (see Fig. 3b) and the closing line is treated as either a free boundary (when it lies outside other parts) or as a merging boundary (when it lies over or under an existing part). Typically, the user draws open curves for the sake of merging into another body part; we focus on this use case for the remainder of the paper.

From these inputs, a set of planar regions is created, each representing a specific model part, for which relative ordering in depth and merging boundaries are known (see Fig. 3c). We merge these input regions to form a joint, possibly non-planar mesh, which is subsequently inflated to have a basic rounded 3D shape (Fig. 3e), and finally deformed (Fig. 3f) to satisfy relative depth ordering of individual parts (Fig. 3d) as well as additional constraints specified by the user via animation handles (Fig. 3g). The user can later assign motion trajectories to these handles in order to animate the inflated 3D mesh.

In the following sections we describe the formulation of our joint framework in the continuous domain and then discuss the details of our discrete implementation.

3.1 Joint formulation

In this section, we describe a continuous formulation of our novel approach, and then in the following sections we develop a corresponding discrete variant and show how to solve it efficiently.

As input, we have a user-drawn curve D_p for each body part p. When D_p is an open curve we automatically close it using a merging curve B_p to delineate the entire part's bounded domain Ω_p, i.e., $\partial\Omega_p = D_p \cup B_p$. The part domains are then "stitched" together at the location of merging curves B_p to form $\Omega = \bigcup_{p=1}^{m} \Omega_p$, the union of all m regions Ω_p. Although such a domain merging step is performed also in the method of Dvorožňák et al. [2018], in our case we aim to produce full 3D models, i.e., we need to duplicate each part's domain to add its back-facing side that is treated differently than the front (this is a non-trivial extension described later in Section 3.2).

In the next step we inflate the Ω, using the method of Sýkora et al. [2014]; i.e., we solve a Poisson equation, $\Delta\tilde{h}(x) = c$, for a height field $\tilde{h}(x) : \Omega \to \mathbb{R}$ jointly over all the parts, subject to Dirichlet boundary conditions along user-drawn contours, $\tilde{h}(x) = 0$ $\forall x \in \{D_p\}$, where Δ is the Laplace operator and c is a user-specified scalar corresponding to a global amount of inflation for each side of the region. When $c < 0$ the inflation goes toward the viewer (front-facing side) and when $c > 0$ it goes further away from the viewer (back-facing side). Since the resulting surface $\tilde{h}(x)$ tends to have a parabolic profile, Sýkora et al. convert it to a more pleasing semi-elliptical height field function h^0 by taking the square root of $\tilde{h}$, i.e., $h^0(x) = \sqrt{\tilde{h}(x)}$.

Note that $h^0(x)$ is essentially a union of connected, overlapping height fields over stitched, overlapping domains $\{\Omega_p\}$, i.e., two height fields per part p (front- and back-facing side). We now treat the resulting union as a 3D surface $g^0 = [x, h^0(x)]$ and we explicitly model its topology to form a manifold. This is an important difference when compared to the method of Dvorožňák et al. [2018] that operates only on a set of front-facing height fields to produce the final bas-relief model.

After the inflation, the resulting shape has smooth connections between parts; however, they interpenetrate heavily, because nothing enforces one part to be in front of another, i.e., there is no depth ordering prescribed. To produce the final consistent surface g we minimize our novel ARAP-L objective:

$$\int_{g^0} \min_{\mathbf{R}\in SO(3)} \|\nabla g - \mathbf{R}\,\nabla g^0\|^2\, ds \tag{1}$$

subject to depth ordering constraints for modeling:

$$\begin{aligned} g_z(s_p) &= g_z(s_q) \quad \forall (s_p, s_q) \in C^{=}, \\ g_z(s_p) &\le g_z(s_q) \quad \forall (s_p, s_q) \in C^{\le}, \\ g_z(s_p) &\ge g_z(s_q) \quad \forall (s_p, s_q) \in C^{\ge}, \end{aligned} \tag{2}$$

and positional constraints added by the user for animation:

$$g(s_i) = \mathbf{p}_i \quad \forall (s_i, \mathbf{p}_i) \in C^{\text{pos}}, \tag{3}$$

where ds is the surface element on g^0; ∇ is the surface gradient operator; $\mathbf{R} \in SO(3)$ is the best-fitting rotation between the initial state and the deformed state for each point on the surface; g_z is the z-coordinate of the resulting model, with z being the direction toward the viewer; $C^{=}$, $C^{\le}$, and $C^{\ge}$ are sets of point pairs that specify relative modeling depth order for two regions; and C^{pos} stands for a set of positional animation constraints. In this case, with s_p and s_q denoting locations on the surface of parts p and q, respectively, an ordering constraint such as $g_z(s_p) \ge g_z(s_q)$ applies if the orthogonal projections along the z-axis of $g_z(s_p)$ and $g_z(s_q)$ have the same 2D coordinates and part p is in front of part q.

A key, novel aspect of our joint ARAP-L objective is that it enables deforming g^0 to simultaneously enforce the layering constraints needed for modeling *and* enable users to animate the model using positional constraints. Moreover, since the layering constraints in Eq. (2) are updated dynamically, *during the animation* regions of the body that are newly covered by a body part begin to deform in response, while other regions, no longer covered, release their deformations. Such a functionality would be difficult to achieve using the method of Dvorožňák et al. [2018] which tends to produce severe distortion artifacts in this interactive scenario (see Fig. 12).

In the next sections, we describe in detail the discrete formulation and implementation of our approach.

3.2 Inflation

In this section we describe the discretization of the inflation step.

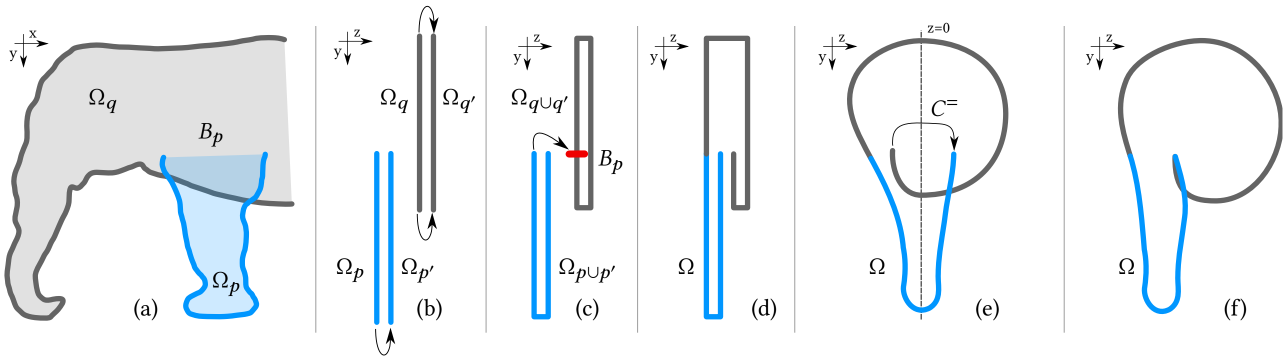

Fig. 4. An example of domain stitching and mesh closure: (a) Two planar domains Ω_p & Ω_q corresponding to a leg part p (drawn as an open curve) and a body part q (drawn as a closed curve) are stitched together through a merging curve B_p. (b) We illustrate a vertical cross-section of the domains, slicing (a) along a yz-plane passing through the body and leg. The two domains Ω_p and Ω_q are duplicated so that their symmetric replicas $\Omega_{p'}$ and $\Omega_{q'}$ lie on the opposite sides of the original domains. (c) Then the symmetric domains Ω_p & $\Omega_{p'}$ are stitched along their boundaries — except along B_p where they remain open — to form $\Omega_{p\cup p'}$, and domain Ω_q is stitched along its entire closed boundary to $\Omega_{q'}$. The domain Ω_q is then split (red line) along B_p on the front side, creating a hole in Ω_q. As drawn, the upper part of the hole is then connected to Ω_p to form one connected domain Ω (d). This single connected domain Ω is then inflated into 3D as a union of front- and back-facing height fields that are topologically connected due to domain stitching and now represent an initial mesh (e). Before deforming this mesh, inequality constraints are constructed to move the front half of the leg to lie in front of the body. To prevent the back half of the leg from popping out of the body during animation, we also add an equality constraint $C^=$ that forces the vertices along the originally open boundary in $\Omega_{p'}$ to agree with the corresponding vertices along the lower half of the hole created in Ω_q. After enforcing these constraints with deformation, the front half of the leg moves to lie in front of the body, and the back half penetrates the body and meets up with the body vertices inside (f). These constraints are updated as needed and satisfied throughout animation.

As noted in Section 3.1, to create a consistent closed model, we generate two regions for each user-drawn part, one facing towards the viewer and one facing away. The region facing the viewer is inflated according to the constant c, while the region facing away is inflated by $-c$. In terms of notation, we treat these inflated regions as separate body parts p and p', corresponding to, e.g., the front half of a leg and the back half of the same leg, respectively. These paired front- and back-facing regions are stitched together along the user-drawn curve D_p (and its identical copy $D_{p'}$).

We convert the continuous, uninflated model Ω into a triangle mesh consisting of vertices $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. Let us consider a single body part p covering the planar region Ω_p defined by a user-drawn contour D_p. As mentioned, if the contour is open, we complete it with an additional boundary curve B_p (e.g., the green line in Fig. 3c). The union of these curves forms $\partial\Omega_p$, sampled to form a fine polygon bounding the region. Then we insert auxiliary vertices into its interior and compute a Delaunay triangulation constrained by the edges along the vertices of $\partial\Omega_p$. We then attach parts by stitching their domains together one-by-one. Suppose part p is attached to part q by open boundary B_p, as shown in Fig. 4a. We insert vertices and edges aligned with B_p into the triangulation of Ω_q, duplicate these inserted vertices, and split the mesh along B_p, creating a "hole" of zero area along B_p in Ω_q (red line in Fig. 4c). We then join the triangulation of Ω_p with Ω_q at B_p along one side of the hole so as to smoothly continue one domain into the other; since we are just working with a planar triangulation, smooth continuation simply means that Ω_p connects to the side of the hole with neighboring vertices along B_p in Ω_q not already covered by Ω_p (Fig. 4d). Note that the resulting stitched triangulation is still manifold, due to the duplication and hole creation procedure.

With the topology of the domain mesh defined, we can now compute the height field $\tilde{\mathbf{h}} = [\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_n]^\top$ over the domain. At each domain vertex i away from user-drawn curves (i.e., at $\mathbf{x}_i \in \Omega_{P_i} \setminus D_{P_i}$, where P_i denotes the body part that vertex i belongs to), $\tilde{h}_i$ follows the Poisson equation:

$$\Delta\tilde{h}_i = \sum_{j\in\mathcal{N}_i} w_{ij}(\tilde{h}_j - \tilde{h}_i) = s_i \cdot a_i \cdot c, \tag{4}$$

where $\mathcal{N}_i$ is the set of vertices in the neighborhood of i, w_{ij} are the standard cotangent Laplace-Beltrami weights [Crane et al. 2013], s_i is $+1$ for a front-facing region and -1 for a back-facing region, and a_i is $1/3$ the sum of the areas of the triangles incident to vertex i.

Conversely, the height $\tilde{h}_i$ of a user-drawn boundary vertex i is subject to the Dirichlet boundary condition:

$$\tilde{h}_i = 0 \quad \forall \mathbf{x}_i \in D_{P_i}\,. \tag{5}$$

Assembling the $s_i \cdot a_i \cdot c$ into vector $\mathbf{c}$, we then solve the Poisson equation across all of the domains as a linear system:

$$\mathbf{L}\tilde{\mathbf{h}} = \mathbf{c}, \tag{6}$$

where $\mathbf{L}$ is the matrix form of the standard cotangent discretization of the Laplace-Beltrami operator, modified to include boundary conditions (5).

To obtain a semi-eliptical shape we compute the height values as:

$$h_i^0 = s_i\sqrt{|\tilde{h}_i|}. \tag{7}$$

Finally, we convert the solution to a 3D mesh with n vertices with each vertex $\mathbf{g}_i^0 = [\mathbf{x}_i, h_i^0]$ and connectivity given by the (stitched) triangulation of all the regions $\{\Omega_p\}$.

Fig. 3e shows a typical model after inflation. Note that regions whose domains were stitched together now join smoothly, e.g., the front side of part A attached to part B in the figure. Front- and back-facing height fields for a given body part (e.g., front and back side of the same leg) are also joined. Due to the Dirichlet boundary conditions, however, the height fields that were stitched together are rooted at the $z = 0$ plane along the user-drawn curves. In the next section, we use as-rigid-as-possible deformation with layering constraints to address this problem, and to add the ability for the user to directly deform and animate the mesh.

3.3 Deformation

The inflation procedure described in the previous section provides us with a surface $\mathbf{g}^0$ that is *locally* well shaped, but whose parts are not properly arranged in space. Therefore, $\mathbf{g}^0$ serves as a reference for a preferred surface curvature, so to speak. To satisfy depth ordering and positional constraints and arrive at a reasonable 3D mesh model, we minimize our ARAP-L objective.

For all initial vertex positions $\mathbf{g}_i^0$ in the inflated model, we seek their final positions $\mathbf{g}_i$ that minimize the following objective:

$$\frac{1}{2}\sum_i \min_{\mathbf{R}_i \in SO(3)} \sum_{j \in \mathcal{N}_i} w_{ij} \|(\mathbf{g}_i - \mathbf{g}_j) - \mathbf{R}_i(\mathbf{g}_i^0 - \mathbf{g}_j^0)\|^2, \tag{8}$$

subject to layering and positional constraints. Here, $\mathbf{R}_i$ is the best fitting rotation between the set of original mesh edges and the unknown (deformed) mesh edges, weighted by the standard cotangent weights w_{ij}.

The layering constraints consist of:

$$\begin{aligned} \mathbf{g}_{i,z} &\geq \mathbf{g}_{j,z} \quad \forall(i,j) \in C^{\geq} \\ \mathbf{g}_{i,z} &\leq \mathbf{g}_{j,z} \quad \forall(i,j) \in C^{\leq} \\ \mathbf{g}_i &= \mathbf{g}_j \quad\ \ \forall(i,j) \in C^{=} \end{aligned} \tag{9}$$

where $C^{\geq}$ and $C^{\leq}$ are the ordering constraints, i.e., constraints on z coordinates that keep parts in front of and behind other parts, respectively, and $C^{=}$ associates vertex pairs that should coincide.

Specifically, if part P_i is in front of part P_j, then $C^{\geq}$ contains (i, j) if i corresponds to a user-drawn boundary vertex with $\mathbf{x}_i \in D_{P_i}$ and $\Pi(\mathbf{g}_j)$ is the nearest point to $\Pi(\mathbf{g}_i)$, where $\Pi(\cdot)$ performs orthographic projection onto the plane $z = 0$. Fig. 3d illustrates this case, with the yellow curves representing boundary points of a part (A) that must stay in front of another part (B). The set $C^{\leq}$ is defined analogously, illustrated by the cyan curve in Fig. 3d. Note that, for efficiency purposes, we only consider the user-drawn curves for the inequality constraints. The remainder of the interior regions tend to move as rigidly as possible according to the movement of these curves, approximately keeping the parts in front (or behind) one another as needed. Enforcing the layering constraints only along drawn curves also has the effect of allowing interpenetration of, for example, the back half of a part through another part it is attached to; we discuss this further below.

Meanwhile, $C^{=}$ may contain the boundary of a part such as, for example, a small spheroid representing the eye of an animal to be attached to corresponding points on the animal's head. In practice, we use the equality constraints to keep the back half of a part from popping out from the interior of the body part it is attached to (see Fig. 4e-f). Note that the animator has the option of disabling these equality constraints to improve performance without compromising quality if the animated motions do not cause a part to pop out.

In addition to these layering constraints, we need to satisfy positional constraints $\mathbf{p}_i$ specified by the user to animate the final mesh:

$$\mathbf{g}_i = \mathbf{p}_i \quad \forall(\mathbf{g}_i, \mathbf{p}_i) \in C^{\text{pos}}. \tag{10}$$

These constraints can be specified in 3D, or, when interaction is strictly in 2D, they can be adjusted to ensure equality only in the xy-coordinates of $\mathbf{g}_i$ and $\mathbf{p}_i$.

Minimizing the ARAP-L objective (8) subject to layering (9) and user-interaction constraints (10) is equivalent to minimizing the following objective (neglecting additive terms not dependent on $\mathbf{g}$):

$$\min_{\mathbf{g}} \ \frac{1}{2}\text{tr}[\mathbf{g}^{\mathsf{T}}\mathbf{L}(\mathbf{g}^0)\mathbf{g}] - \text{tr}[\mathbf{R}(\mathbf{g}^0, \mathbf{g})\mathbf{K}(\mathbf{g}^0)\mathbf{g}], \tag{11}$$

subject to linear (in)equality constraints:

$$\begin{aligned} \mathbf{A}_{\text{ieq}}\mathbf{g}_z &\leq \mathbf{b}_{\text{ieq}}\,, \\ \mathbf{A}_{\text{eq}}\mathbf{g} &= \mathbf{b}_{\text{eq}}. \end{aligned} \tag{12}$$

In the main objective (11), $\mathbf{L}(\mathbf{g}^0)$ is the standard cotangent discretization of the Laplace-Beltrami operator with weights computed over the source mesh, $\mathbf{R}(\mathbf{g}^0, \mathbf{g}) = (\mathbf{R}_1, \ldots, \mathbf{R}_n) \in SO(3)$ is a matrix of the stacked local rotations (whose value depends on the unknown $\mathbf{g}$ as well as fixed $\mathbf{g}^0$), and $\mathbf{K}(\mathbf{g}^0) \in R^{3n\times n}$ stacks differential coordinates of the source shape $\mathbf{g}^0$. A more detailed derivation of this matrix format of the ARAP energy is given in [Jacobson et al. 2012] (Eq. (11) in their paper).

For the (in)equality constraints (12), $\mathbf{A}_{\text{ieq}}$ and $\mathbf{A}_{\text{eq}}$ are matrices of linear coefficients for (in)equality and positional constraints (sparse matrices used to take simple differences between coordinates), and $\mathbf{b}_{\text{ieq}}$ and $\mathbf{b}_{\text{eq}}$ are the right-hand-side vectors of constraints. Note that $\mathbf{A}_{\text{eq}}$ includes equality constraints from both $C^{=}$ and C^{pos}. In the examples in this paper $\mathbf{b}_{\text{ieq}} = \mathbf{b}_{\text{eq}} = \mathbf{0}$, though these right-hand-side vectors could in principle be non-zero if the user wished, e.g., to bring an overlapping surface further to the fore.

We solve the optimization problem (11)-(12) using the iterative local-global approach of Sorkine and Alexa [2007] combined with an active-set method [Nocedal and Wright 2006]. In each iteration, we first fit the local rotations $\mathbf{R}(\mathbf{g}^0, \mathbf{g})$ for the current estimate of the value of $\mathbf{g}$ using SVD on each mesh 1-ring. We then consider these rotations fixed, which turns the objective in (11) into a quadratic expression in $\mathbf{g}$, so that (11)-(12) becomes a quadratic programming (QP) problem. We solve the QP using active set, which enables us to control the number of performed iterations so as to maintain an interactive frame rate. We alternate between locally fitting the rotations and solving the QP in a continuous loop, where the constraints (12) are continuously updated according to user interaction.

Fig. 3f shows the result of applying the ordering constraints to a model. The boundaries of legs now correctly shift off of the $z = 0$ plane (pulling the interiors with them) to satisfy the constraints. As noted above, we see that, e.g., the back half of a leg that sits in front of the torso is not constrained to be in front of the torso. We also tried imposing this constraint, which would move the entire leg in front of the torso in this case. We found that both looked reasonable, but that the interpenetrating solution still gave a good sense of part attachment while running faster due to fewer constraints. Finally, with our ARAP-L framework in place, the user can also deform and animate the model with point constraints, reposing the model as shown in Fig. 3g.

4 USER EXPERIENCE DESIGN

In designing the user experience for Monster Mash, we focused on certain qualities with the intention of maximizing the user's creativity:

- *Responsive*: actions have immediate perceivable consequences.
- *Fluid*: dynamic actions have dynamic results.
- *Intuitive*: behaves in an expected and predictable way.
- *Discoverable*: you can learn how it works by playing with it.
- *Low-risk*: it takes minimal effort to try out a new idea, so failure is not costly.

Our tool has two main interaction modes: a *drawing mode*, in which the user creates sketches by drawing (as described in Sec. 3), and an *animation mode*, where the user can move and animate the resulting 3D mesh. The user can switch back and forth between these two modes at will, and as much information as possible is preserved across those transitions. For instance, if the user animates all of a character's limbs, and then later goes back to drawing mode and deletes a limb, the animation on the remaining parts will be preserved. This gives the user freedom to experiment with different designs and body plans, reduces the cost of mistakes, and encourages a deeper understanding of the relationship between the character's design and its animation.

Previous tools based on ARAP deformation [Jacobson et al. 2012; Sorkine and Alexa 2007] allowed the computation to converge to a final result before each frame is drawn. In our tool, we instead let the user observe the individual iterations as an ongoing dynamic process, creating a compelling illusion that the character is made of some flexible material with its own physical properties. Users have described it as feeling like an inflatable toy or a stuffed animal. This makes the interaction more enjoyable, and also makes the character's behavior under deformation a discoverable quality, which is important for developing the user's intuition.

In the animation mode, the user can create a control point anywhere on the character's surface, and animate it by moving it in real time while the software records the motion. The user can create and animate additional control points as desired, and their movements will then be automatically synchronized with the timing of the first (master) point. In this way the user can build up a complex action like a walk by layering cycles, one body part at a time.

Coordinating movement between multiple control points in real time is not a trivial task — it requires some skill on the user's part, like learning to dance. The Monster Mash tool provides a few options to make this skill easier to master. There are three different recording modes, designed to solve the synchronization problem in different ways: (1) overwriting fixed-length cycles, where the control point's movement is recorded continuously in a loop, constantly overwriting the previous cycle; (2) averaging fixed-length cycles, where the current movement is blended with previous cycles, which can serve to smooth out kinks and pops; and (3) time-scaled cycles, where each cycle begins on the same frame, but all the cycles are stretched or compressed in time to match the timing of the master control point. There is also an optional metronome, which provides an audio "click track" to help the user get into the rhythm of the movement.

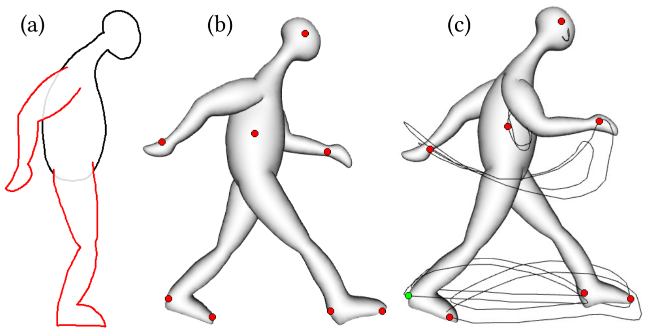

Fig. 5. A walking cycle sequence: (a) input hand-drawn sketch, (b) inflated 3D model with control points, (c) walking cycle animation created by recording trajectories of individual control points specified by the user.

Each control point's motion is represented visually by a curve in screen space. The user can copy and paste these curves from one control point to another, and adjust their overall position, orientation, and scale. The user can also speed up, slow down, or offset each point's timing using hotkeys. So, for instance, if the user wants a perfectly symmetrical walk, she can copy the motion from the character's left foot to the right foot and offset the timing by half a cycle (see Fig. 5 and our supplementary video).

5 RESULTS

We implemented our approach using C++, libIGL[1], Eigen[2], and, for the user interface, Qt[3]. To achieve interactive performance, we paid special attention to the difficulty of finding correspondences between mesh vertices needed to satisfy the relationships in equation (12). To avoid on-the-fly re-meshing, we refine the model to have a sufficiently dense sampling of vertices. We then maintain a uniform spatial subdivision in 2D, with each cell containing a set of vertices with similar (x, y) coordinates. Using this data structure, we can quickly retrieve corresponding candidates for a given (in)equality constraint. Due to the layering of parts, a query vertex in one layer ℓ may have corresponding vertices across multiple other layers. In this case, we choose only the correspondences in the two layers on either side of ℓ that are closest with respect to the total depth ordering of individual model components. This approach avoids creating a linear dependence of constraints that could lead to an over-constrained linear system when optimizing (11). Using this particular implementation of handling vertex correspondence on a single-core 3GHz CPU, a model consisting of 6k vertices and 12k faces can be manipulated at interactive rates: around 7 frames per second.

To further increase the frame rate we allow the user to optionally switch between two interaction modes: (1) a full-fledged per-frame

[1]https://libigl.github.io/
[2]http://eigen.tuxfamily.org/
[3]https://www.qt.io/

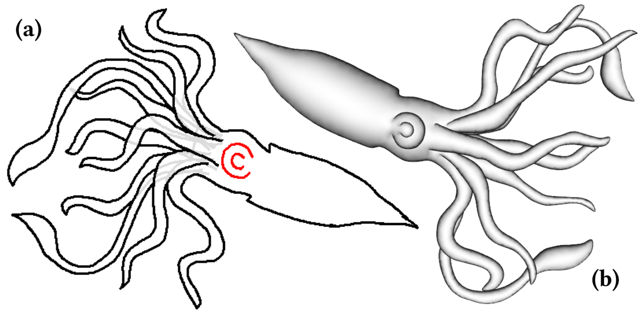

Fig. 6. Our approach can handle 3D models with complex topology containing an arbitrary number of components stored in multiple overlapping layers (b). Strokes that specify boundaries of these components can freely overlap (a).

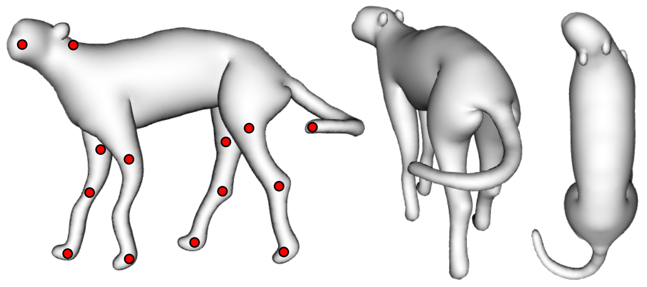
Fig. 7. Although our framework was primarily developed for a single viewpoint workflow, it also allows for out-of-plane rotations. The assumption is that these user edits will not be too excessive, as, in extreme cases, out-of-plane rotations can invalidate depth-ordering constraints.

computation of the ARAP-L objective; and (2) interleaved processing, where two threads are running in parallel, with one computing simple ARAP deformation without layering constraints, and the other providing a full-fledged solution using the ARAP-L formulation. In this latter case, xy-coordinates for a vertex are taken from the ARAP thread (updated quickly) and are combined with the z-coordinate taken from the ARAP-L thread (updated more slowly). With this interleaving, we can increase the frame rate to around 15 frames per second, which significantly improves responsiveness of the whole system and makes the interaction more enjoyable.

Our system has no specific limits on the number of individual components or layering complexity. It can handle sketches that contain multiple overlapping strokes (see Fig. 6). Although its primary use case is the single-view scenario where we deliberately limit rotations $\mathbf{R}_i$ in (8) to stay in the xy-plane, our framework can also handle out-of-plane manipulations by allowing $\mathbf{R}_i$ to include full 3D rotations (see Fig. 7) provided that they not violate the depth-ordering constraints (9) specified with respect to the original viewpoint.

We evaluated our approach on a set of organic models. Results are presented in Figures 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, and in our supplementary materials. In each caption, we refer to our "inflated model" as the model after inflating the drawing and satisfying the ordering constraints, but before the user has begun editing and animating

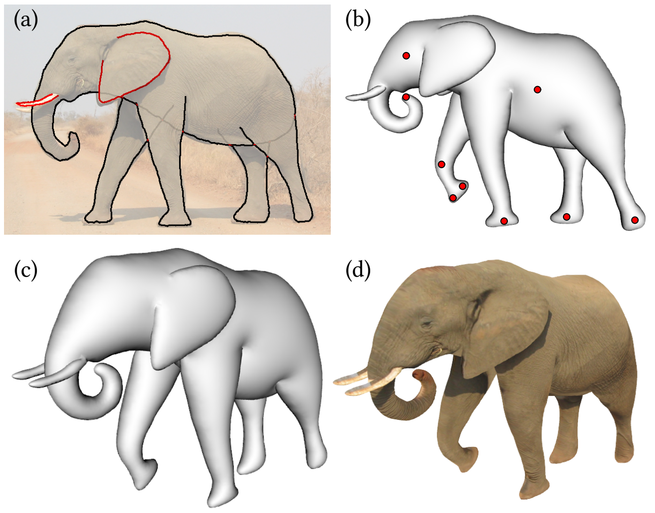

Fig. 8. Elephant result: (a) input hand-drawn sketch traced over the photograph; (b) inflated 3D model deformed to a new pose using a set of control points specified by the user (red dots); (c) the deformed model viewed from a different viewpoint; (d) the inflated model with a texture taken from the original photo. Source photograph "Elephant side-view Kruger.jpg" by Felix Andrews / CC-BY-SA-3.0.

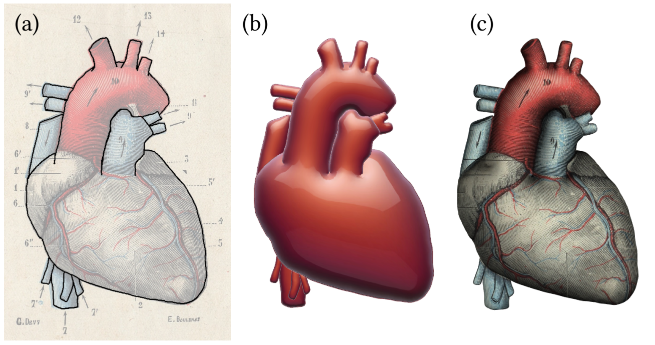

Fig. 9. Heart result: (a) original scientific illustration of the heart with a set of input hand-drawn strokes traced over the illustration; (b) inflated 3D model; (c) inflated model with a texture taken from the original illustration and shading.

the 3D model. When a model is traced over an existing photograph, the original texture from the photograph can be mapped onto it to produce a photorealistic look (see Figures 8 and 9). In particular, we simply map the xy-coordinates of mesh points directly to uv-coordinates of the image — a simple, orthographic "x-ray" projection texture mapping. For purely hand-drawn models a more stylized appearance may be preferred. In this case we can apply real-time, example-based stylization [Sýkora et al. 2019] (see Fig. 10).

The presented results demonstrate that only a limited amount of user interaction is required to produce relatively complex 3D models and to immediately animate them. The resulting animations are free of artifacts that can otherwise arise when not adapting the original shape to a new pose (see Fig. 2). By taking into account

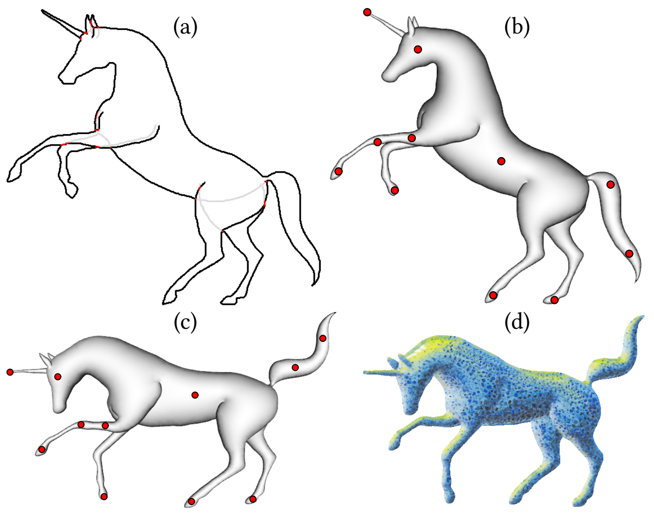

Fig. 10. Unicorn result: (a) input hand-drawn sketch; (b) inflated 3D model with a set of control points specified by the user (red dots); (c) a new pose of the model specified by moving the control points; (d) stylized render of the deformed model seen from behind.

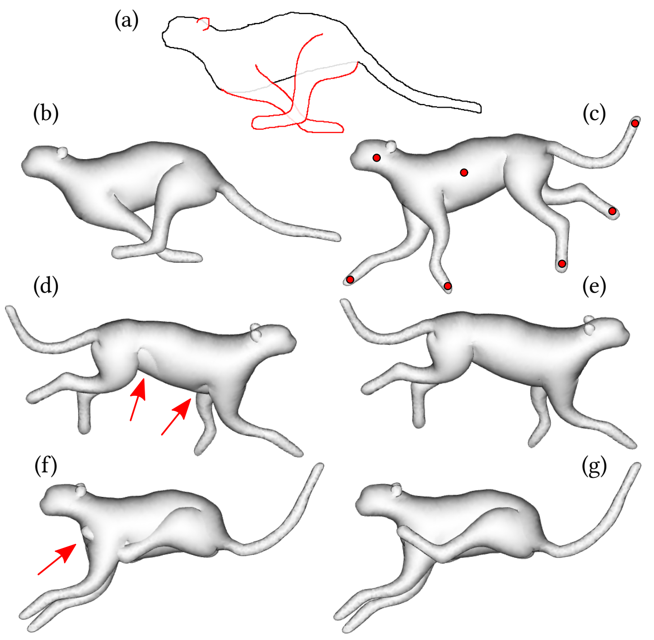

Fig. 11. Cheetah result: (a) input hand-drawn sketch; (b) inflated 3D model with a set of control points specified by the user (red dots); (c) a new pose of the model specified by moving the control points. (d) When shape-preserving deformation is applied directly on (and "baked into") the inflated mesh, shape modeling artifacts become visible when re-posing (red arrows). (e) Our approach can adapt the model to respect the new pose and thereby suppress such artifacts. When the inflated 3D model is manipulated using shape-preserving deformation from a side view (f), some parts of the model may collide and penetrate (red arrow). This is caused by the fact that the absolute position in depth is difficult to specify from this viewpoint. Our framework resolves this problem automatically (g) by satisfying prescribed relative depth-ordering constraints specified during the sketching phase.

relative depth-ordering constraints our technique can automatically avoid collisions and allow the user to animate the model in the 2D plane without having to change viewpoint to resolve collisions (see Figures 2 and 11). Besides automatic shape adaptation and collision handling, in motion our framework also avoids the sort of unnaturally stiff deformations common in, for example, skinning frameworks. Instead, it creates a lively dynamic behavior that gives the feel of a flexible material, for which a complex simulation of an accurate physical model would otherwise be required. On top of this secondary motion, additional procedurally generated dynamic effects can be added by manipulating the rotation matrices $\mathbf{R}_i$ in (8), e.g., changing their scaling factor to simulate breathing (see our supplementary video).

Our approach can also be used to create simple animation cycles. Thanks to our animation interface described in Section 4 the user can interactively move individual control points and record their closed trajectories in time. Those can then be replayed in real time to produce the final animation (see walking cycle example in Fig. 5 and our supplementary video).

5.1 Users' feedback

We conducted an informal study to evaluate our user interface proposed in Section 4 as well as to get a general feedback from prospective users. We gave our system to a group of 2 novice users and 5 artists with various levels of experience in 2D and 3D computer animation (80%) as well as traditional hand-drawn animation (60%). Their task was to prepare a 3D animation in their own style from scratch. The introductory video we provided them as well as results of their efforts can be viewed in Fig. 1 and in our supplementary video.

Artists as well as novice users had fun playing with the system and appreciated how quickly their hand-drawn characters could be converted into a 3D model and immediately brought to life. They liked the possibility of obtaining appealing results instantly. The system also motivated them to be more spontaneous. They also found an interesting added value of the system — the dynamic effect that resembles momentum or gravity, and which introduces a secondary motion superimposed on their own animation. This effect can help give the appearance of more complicated natural motion without the need to animate it explicitly or to use any sort of complicated keyframe-based solution.

The most common suggestions given by artists were to add a timeline and use keyframes for finer control over the timing of the animation loops. They also would appreciate having greater control over the stiffness and softness of different parts of the character. One artist was very much interested in having a motion-parenting option, e.g., to be able to design an animation of wings on a bee, and then animate the bee with the wings moving. Some artists found the animation they were creating hard to predict, which may be fun when exploring ideas, but can be a drawback in a more serious context where highly accurate timing and positioning is required.

Novice users were excited they can create and animate a relatively complex 3D model on their own. Monster Mash was a first tool that allowed them to accomplish such a seemingly difficult task they initially perceived as almost impossible. This positive experience motivated them to explore the system more. Although they were not able to use the tool to a similar extent as professional artists, they enjoyed the possibility of using only a single control point with which it is already feasible to generate relatively complex motions for their characters.

The overall outcome of our informal study can be summarized in an interesting reaction of one professional animator: "I don't see Monster Mash as a replacement for traditional 3D animation tools, but almost like a different medium: a new way to animate characters that combines elements of design, sculpture, animation, and puppetry."

5.2 Comparison

We compared our approach to RigMesh [Borosán et al. 2012], which is used as the modeling and rigging part of the AniMesh system [Jin et al. 2015]. It represents the current state of the art in systems that combine sketch-based modeling with animation. As shown in the supplementary video as well as in Fig. 2, RigMesh requires far more manual intervention than our system to create a model (e.g., to model the mouse in Fig. 2, our approach took 1 minute whereas RigMesh required almost 7 minutes). Moreover, RigMesh relies on positioning in a 3D space as well as on careful planning of the skeletal structure. In our tool, after drawing a few strokes the model is immediately ready for animation. In addition, during the animation with RigMesh the model needs to be observed from different viewpoints to avoid interpenetrations. In our system collisions are automatically avoided thanks to depth-ordering constraints. Also, the shape produced by RigMesh looks unnatural due to visible bulges produced by Laplacian smoothing at the boundaries of individual parts (see our supplementary material for additional comparison with RigMesh).

We also compared our results with the models produced by the method of Dvorožňák et al. [2018]. A key difference here is that in our case we can produce full 3D model (Fig. 12). Moreover, when their bas-relief approximation is animated using their deformation model, which is not shape-preserving, the resulting mesh suffers from severe distortion artifacts (c.f. Fig. 12 and the supplementary video). For comparisons of computational overhead and results of other sketch-based modeling systems, including the methods of Entem et al. [2015; 2019], Bessmeltsev et al. [2015], and Li et al. [2018], please refer to our supplementary meterial.

6 LIMITATIONS AND FUTURE WORK

Although we demonstrated how our method makes it possible to rapidly create consistent animated meshes from a few hand-drawn strokes, there are some limitations, which may motivate follow-up work.

One of the key trade-offs of our single-view modeling scenario is that the depth-ordering constraints are related to the original viewpoint. This fact inherently limits the extent to which the model can be deformed, as a sufficiently large distortion may change the pose

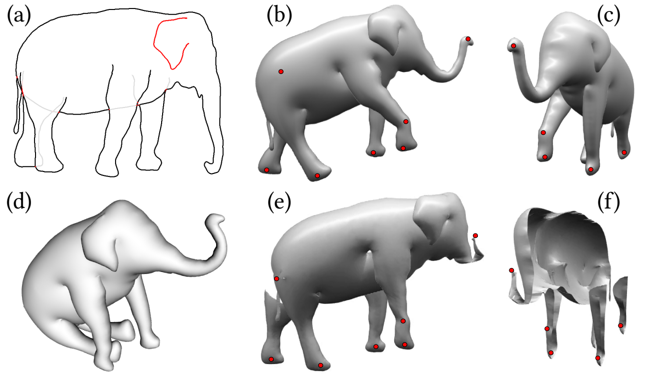

Fig. 12. Comparison with Dvorožňák et al. [2018]: the input sketch (a) was used to inflate and deform the model of an elephant using our approach (b, c) and also using the method of Dvorožňák et al. (e, f). Note, that our approach inflates a complete 3D mesh, whereas the method of Dvorožňák et al. produces only its bas-relief approximation as seen in the side view (f). In addition, when positional constraints are added to the framework of Dvorožňák et al., the resulting deformation does not preserve the original shape well and produces unpleasant distortions (e). A more complicated deformation such as (d) would be impossible to achieve using approach of Dvorožňák et al.

in a valid way that conflicts with the original depth ordering. Thus, a more flexible ordering, such as ordering related to the direction of the surface normal, may be more appropriate. Similarly, the user might intentionally deform individual model parts in such a way that their depth ordering would change. In this case an automatic mechanism could be developed to recognize such an event and change the depth-ordering constraints accordingly (as done, e.g., in [McCann and Pollard 2009]). This extension is a natural direction for future work.

Another common drawback of the single-view modeling scenario is the limited control over the proportions in depth. Although inflation together with the depth-ordering constraints usually leads to good depth proportions, sometimes additional control from a different viewpoint would be beneficial (e.g., when trying to give more depth to a nose by pulling the tip outward in z in a frontal view of a face). As future work we envision an extension of our framework that would enable merging input from multiple different poses or viewpoints. This could be beneficial, for example, in the context of modeling from traditional animation where a sequence of hand-drawn frames could be used as a reference for the resulting 3D model.

Although our framework provides interactive response for models of moderate size, we would like to improve it further so that larger meshes could also be edited at interactive rates. One possibility for reducing computational overhead is to employ a multi-scale strategy, solving on a low-resolution mesh, and then iteratively upsampling and solving on higher resolutions until converging to the optimal solution.

When texturing a shape with the reference imagery used to guide sketching, we apply a very simple orthographic "x-ray" projection scheme to assign texture coordinates. Texturing occluded regions could be improved with part-aware texture synthesis, but the general task of inferring the appearance of hidden regions is an open problem, currently an active topic of investigation in the deep learning community. In addition, image pixels around the silhouettes of objects are stretched onto the surface and could be improved; similarly, the surface itself could be refined to yield triangles with better aspect ratios that would improve shading even without texture.

7 CONCLUSION

We have presented a joint approach to sketch-based modeling and animation. Our technique enables quick 3D model inflation from a sparse set of user-specified strokes, which can then be instantly animated without needing to specify an explicit skeletal structure or preparing a custom deformation rig. In contrast to previous works that structured modeling and deformation as two separate tasks, our method treats deformation as an integral part of the modeling process. This unification allows the inflated model to better accommodate user-specified deformations. We evaluated our approach on a variety of 3D models and compared the results with the current state of the art. In contrast to previous approaches, comparably complex 3D animations can be created using only a few strokes and control points—and in a fraction of the time.

ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments and suggestions. We are also grateful to Hélène Leroux, Neth Nom, David Murphy, Samuel Leather, Pavla Sýkorová, and Jakub Javora for participating in the interactive sessions. This research was supported by Google, the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/179/OHK3/3T/13 (Research of Modern Computer Graphics Methods), and by the Research Center for Informatics, grant No. CZ.02.1.01/0.0/0.0/16_019/0000765.

REFERENCES

Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loïc Barthe. 2008. Matisse: Painting 2D regions for Modeling Free-Form Shapes. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. 57–64.

Mikhail Bessmeltsev, Will Chang, Nicholas Vining, Alla Sheffer, and Karan Singh. 2015. Modeling Character Canvases from Cartoon Drawings. *ACM Transactions on Graphics* 34, 5 (2015), 162.

Cédric Bobenrieth, Frédéric Cordier, Arash Habibi, and Hyewon Seo. 2020. Descriptive: Interactive 3D Shape Modeling from A Single Descriptive Sketch. *Computer-Aided Design* 128 (2020), 102904.

Peter Borosán, Ming Jin, Doug DeCarlo, Yotam Gingold, and Andrew Nealen. 2012. RigMesh: Automatic Rigging for Part-Based Shape Modeling and Deformation. *ACM Transactions on Graphics* 31, 6 (2012), 198.

Frederic Cordier, Hyewon Seo, Jinho Park, and Jun yong Noh. 2011. Sketching of Mirror-Symmetric Shapes. *IEEE Transactions on Visualization and Computer Graphics* 17, 11 (2011), 1650–1662.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 Courses*.

Mihaly Csikszentmihalyi. 1991. *Flow: The Psychology of Optimal Experience*.

Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A. Efros, and Adrien Bousseau. 2018. 3D Sketching Using Multi-View Deep Volumetric Prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 21.

Marek Dvorožňák, Saman Sepehri Nejad, Ondřej Jamriška, Alec Jacobson, Ladislav Kavan, and Daniel Sýkora. 2018. Seamless Reconstruction of Part-Based High-Relief Models from Hand-Drawn Images. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. 5.

Even Entem, Loïc Barthe, Marie-Paule Cani, Frederic Cordier, and Michiel van de Panne. 2015. Modeling 3D animals from a side-view sketch. *Computers & Graphics* 46 (2015), 221–230.

Even Entem, Amal dev Parakkat, Loïc Barthe, Ramanathan Muthuganapathy, and Marie-Paule Cani. 2019. Automatic Structuring of Organic Shapes from a Single Drawing. *Computers & Graphics* 81 (2019), 125–139.

Yotam Gingold, Takeo Igarashi, and Denis Zorin. 2009. Structured Annotations for 2D-to-3D Modeling. *ACM Transactions on Graphics* 28, 5 (2009), 148.

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: A Sketching Interface for 3D Freeform Design. In *SIGGRAPH Conference Proceedings*. 409–416.

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-Rigid-As-Possible Shape Manipulation. *ACM Transactions on Graphics* 24, 3 (2005), 1134–1141.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Transactions on Graphics* 31, 4 (2012), 77.

Ming Jin, Daniel Gopstein, Yotam I. Gingold, and Andrew Nealen. 2015. AniMesh: Interleaved Animation, Modeling, and Editing. *ACM Transactions on Graphics* 34, 6 (2015), 207.

Pushkar Joshi and Nathan A. Carr. 2008. Repoussé: Automatic Inflation of 2D Artwork. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling*. 49–55.

Olga A. Karpenko and John F. Hughes. 2006. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics* 25, 3 (2006), 589–598.

Zohar Levi and Craig Gotsman. 2013. ArtiSketch: A System for Articulated Sketch Modeling. *Computer Graphics Forum* 32, 2pt2 (2013), 235–244.

Changjian Li, Hao Pan, Yang Liu, Alla Sheffer, and Wenping Wang. 2018. Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling. *ACM Transactions on Graphics* 37, 6 (2018), 238.

Chang-Jian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2017. BendSketch: Modeling Freeform Surfaces Through 2D Sketching. *ACM Transactions on Graphics* 36, 4 (2017), 125.

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. In *Proceedings of International Conference on 3D Vision*. 67–77.

James McCann and Nancy S. Pollard. 2009. Local layering. *ACM Transactions on Graphics* 28, 3 (2009), 84.

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Transactions on Graphics* 26, 3 (2007), 41.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*.

Luke Olsen, Faramarz Samavati, and Joaquim Jorge. 2011. NaturaSketch: Modeling from Images and Natural Sketches. *IEEE Computer Graphics and Applications* 31, 6 (2011), 24–34.

Saulo Ramos, Diogo Fernando Trevisan, Harlen Costa Batagelo, Mario Costa Sousa, and João Paulo Gois. 2018. Contour-aware 3D reconstruction of side-view sketches. *Computers & Graphics* 77 (2018), 97–107.

Alec Rivers, Takeo Igarashi, and Frédo Durand. 2010. 2.5D Cartoon Models. *ACM Transactions on Graphics* 29, 4 (2010), 59.

Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A. Jorge. 2005. ShapeShop: Sketch-based Solid Modeling with BlobTrees. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. 53–62.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.

Daniel Sýkora, Ondřej Jamriška, Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, and Eli Shechtman. 2019. StyleBlit: Fast Example-Based Stylization with Local Guidance. *Computer Graphics Forum* 38, 2 (2019), 83–91.

Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters. *ACM Transactions on Graphics* 33, 2 (2014), 16.

Chiew-Lan Tai, Hongxin Zhang, and Jacky Chun-Kin Fong. 2004. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum* 23, 1 (2004), 71–83.

Bui Minh Tuan, Junho Kim, and Yunjin Lee. 2015. 3D-look Shading from Contours and Hatching Strokes. *Computers & Graphics* 51, C (2015), 167–176.

Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F. Hughes. 2007. A Sketch-Based Interface for Clothing Virtual Characters. *IEEE Computer Graphics and Applications* 27, 1 (2007), 72–81.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization. *ACM Transactions on Graphics* 33, 4 (2014), 131.

Chih-Kuo Yeh, Shi-Yang Huang, Pradeep Kumar Jayaraman, Chi-Wing Fu, and Tong-Yee Lee. 2017. Interactive High-Relief Reconstruction for Organic and Double-Sided Objects from a Photo. *IEEE Transactions on Visualization and Computer Graphics* 23, 7 (2017), 1796–1808.