SecondSkin: Sketch-based Construction of Layered 3D Models

Chris De Paoli Karan Singh
University of Toronto

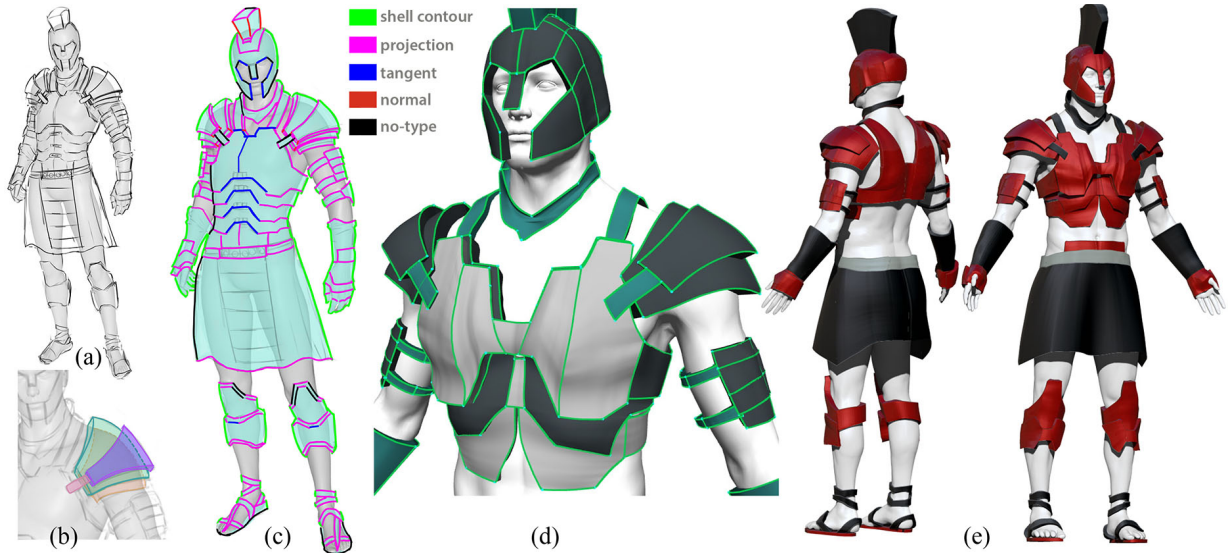


Figure 1: *2D strokes sketched on and around 3D geometry form the input to SecondSkin (a). Layered structures are represented as solid models with volumes bounded by surface patches and curves (b). A majority (91%) of sketch strokes are perceived by viewers as one of four curve-types (c). We automatically classify these strokes based on the relationship between 2D strokes and underlying 3D geometry, producing 3D curves, surface patches, and volumes (d), resulting in layered 3D models suitable for conceptual design (e).*

Abstract

SecondSkin is a sketch-based modeling system focused on the creation of structures comprised of layered, shape interdependent 3D volumes. Our approach is built on three novel insights gleaned from an analysis of representative artist sketches. First, we observe that a closed loop of strokes typically define surface patches that bound volumes in conjunction with underlying surfaces. Second, a significant majority of these strokes map to a small set of curve-types, that describe the 3D geometric relationship between the stroke and underlying layer geometry. Third, we find that a few simple geometric features allow us to consistently classify 2D strokes to our proposed set of 3D curve-types. Our algorithm thus processes strokes as they are drawn, identifies their curve-type, and interprets them as 3D curves *on* and *around* underlying 3D geometry, using other connected 3D curves for context. Curve loops are automatically surfaced and turned into volumes bound to the underlying layer, creating additional curves and surfaces as necessary. Stroke classification by 15 viewers on a suite of ground truth sketches validates our curve-types and classification algorithm. We evaluate SecondSkin via a compelling gallery of layered 3D models that would be tedious to produce using current sketch modelers.

ACM Reference Format
De Paoli, C., Singh, K. 2015. SecondSkin: Sketch-Based Construction of Layered 3D Models. ACM Trans. Graph. 34, 4, Article 126 (August 2015), 10 pages. DOI = 10.1145/2766948 http://doi.acm.org/10.1145/2766948.

SIGGRAPH '15 Technical Paper, August 09 – 13, 2015, Los Angeles, CA.

DOI: http://doi.acm.org/10.1145/2766948

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, and systems

Keywords: sketch-based modeling, layers, shells

1 Introduction

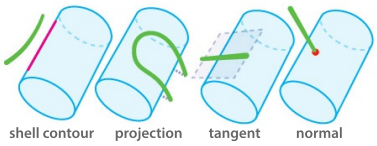
Current 3D conceptual design tools, regardless of being based on metaphors of sketching and sculpting or traditional CAD modeling, typically focus on the creation of the *skin* or visible surface of 3D objects. Physical objects, both organic and man-made, however, are often layered assemblies: comprising parts segmented by form, function or material, built over each other (Figure 1, 2 and 13). While research in character skinning and animation has noted the importance of conceptual anatomic layers for a quarter century now [Chadwick et al. 1989], 3D conceptual design tools, to date, have largely ignored what lies beneath the skin. *SecondSkin* addresses this problem: *the fluid sketch-based creation of layered 3D structures.*

A defining aspect of layered modeling is the geometric dependence of layers on underlying layers. This is clearly evidenced in a multitude of books and tutorials on sketching and concept art [Davies and Scott 2012], where maquettes of underlying layers are used as a visual reference **on** and **around** which to draw subsequent layers (Figure 2). Prior work in sketch-based modeling [Kara and Shimada 2007; Nealen et al. 2007; Takayama et al. 2013], typically interprets sketched strokes as lying **on** template objects or the evolving 3D geometric skin of the object. In the context of layered modeling however, we expect that sketched strokes are largely drawn **around** underlying template objects, to build new layered structures.

While projecting a 2D stroke drawn from a given view **on** to 3D geometry is mathematically precise and straightforward, inferring a 3D curve from such a 2D stroke to lie **around** 3D geometry, is generally

Figure 2: *2D concept art for armour variations drawn over a 3D character,* ©*Paul Richards.*

ill-defined. Despite this, design strokes representing layered geometry trigger 3D percepts that are consistently imagined by viewers (Figure 1(c)). Recently, the formulation of a number of perceived 3D relationships between connected 2D strokes have been exploited to lift design sketches off the page into 3D [Xu et al. 2014]. Complementary to these relationships and perhaps more important for layered modeling, are the perceived 3D relationships between a 2D stroke and the 3D geometry of the layer over which it is drawn. We discover through conversations with artists and analysis of layered sketches (Figures 1(b) and 10), that a majority of design strokes (286 of 313 strokes in Figure 1(c)), $\approx 91\%$, are perceived in relation to the underlying layer geometry, as one of four 3D curve-types shown inset: *shell contour*, *shell projection*, *tangent plane* and *normal plane*. These curve types have individually seen use in sketching interfaces, for example, contours for model deformation [Zimmermann et al. 2007] or normal curves for 3D painting [Schmid et al. 2011]. Together however, they capture the bulk of design strokes drawn to depict layered structures. Strokes that do not belong to these categories, either do conform to them from a different viewpoint, or can be created from context, by anchoring them to strokes of the above curve-types interpreted in 3D. We further find that these 2D strokes can be robustly classified as one of the 3D curve-types, by simply observing the 2D spatial relationships of the curve relative to the underlying geometry. While the classification is not infallible (Figure 11), it is predictable and easily understood by the user, allowing them to redraw strokes differently without frustration when the inference is undesirable.



Once classified, the curve-type defines how the 2D stroke should be interpreted in 3D. Points on a *shell contour* for example, typically lie above their corresponding 3D contour point on the underlying geometry at a height equal to their projected distance in 2D. The general viewpoint assumption that the 2D stroke is strongly indicative of its 3D shape [Nakayama and Shimojo 1992; Xu et al. 2014], provides us with a good metric to evaluate such a 3D interpretation of a stroke. Incorrectly classified strokes, typically result in 3D curves of poor quality. We default these strokes to projections on 3D planes commonly employed in sketch-based modeling [Bae et al. 2008; Schmidt et al. 2009]. These insights form the foundation of our novel 2D stroke to 3D curve inference algorithm (Section 3).

Layered modeling also transcends visible surface modeling to produce 3D solid models comprising closed volumes, bounded by surfaces and curves, an area thus far dominated by traditional CAD based solid modeling techniques. We note that designers often draw a number of curves in a closed loop that defines a *shell-like* surface patch implicitly *sandwiching* a volume with the underlying geometric layer (Figure 1(b)). We are thus able to infer and create 3D volumes along with their bounding surfaces and curves automatically, from a few closed design strokes (Figure 1(d). These 3D volumes, bounded by 4-sided spline patches are easily represented using hexahederal elements suitable for further physical deformation and simulation.

SecondSkin is to the best of our knowledge, the first sketch-based system focused on the creation of layered 3D solid models. Our technical contribution is a principled approach to the interactive inference of layered structures sketched over existing geometry. We spoke to artists, analyzed sketches, hypothesized and perceptually validated curve-types that describe a majority of the strokes sufficient to create compelling 3D models. Specifically, we formulate a set of curve-types commonly used to depict layered structures and an algorithm that infers 3D curves from sketched 2D strokes and their curve-type (Section 3). Our implementation uses a novel drawing workflow that eases the creation of curves, surfaces and volumes (Section 4). We validate our approach with a perception study, where our algorithmic output matches the curve-type classification of 15 viewers over 40 strokes from 5 example sketches (Section 5). We report results and user experiences of artists working with SecondSkin and discuss limitations and future work (Section 6).

2 Related Work

We broadly classify prior art as pertaining to sketch-based or layered and solid modeling, all of which can be traced half a century back to the seminal system Sketchpad [Sutherland 1964].

Layered and Solid 3D Modeling
Research in 3D solid modeling has been dominated by downstream applications of physical simulation and manufacturing, where the need for watertight 3D volumes is more critical than in 3D conceptual design [Shapiro et al. 2001]. 3D solid modelers in research and industry, thus tend to be engineer-centric interfaces, where procedural volumes, with multiple material layers [Cutler et al. 2002], can be combined using constructive solid geometry (CSG). Real-time simulation and the growing ease of 3D fabrication however, motivates the need to bridge the gap between 2D concept art and 3D solid modeling, as addressed by SecondSkin. Implicit surface sculpting techniques [Bloomenthal and Wyvill 1997], while not ideally suited to the creative sketch workflow of layered 3D models like Figure 1 and 2, are complementary in their design support of solid amorphous forms. Research on part-based modeling is closer in spirit to our work [Schmidt and Singh 2008; Schmidt and Singh 2010], allowing mesh parts to be layered and combined together by interactively sliding them over each other to produce a single composite 3D mesh. Interactive layer manipulation for deformable objects has also been explored [Igarashi and Mitani 2010].

While the explicit use of volumetric layers in 3D conceptual design research is rare, a layered approach to 3D character animation is well established [Chadwick et al. 1989], where layers for bone, muscle, skin and clothing influence the dynamic shape of each subsequent layer. Character animation research has looked at the creation of layered structures, both inward (fitting muscle primitives [Pratscher et al. 2005; Vaillant et al. 2013]) and outward (designing and draping clothing [Volino and Magnenat-Thalmann 2000]) from an input skin.

Sketch-based 3D Modeling
There is a large body of work in sketch-based modeling, surveyed by Olsen et al. [2009]. One could categorize these modeling systems as single-view (akin to traditional pen on paper) eg. [Schmidt et al. 2009; Andre and Saito 2011; Chen et al. 2013; Xu et al. 2014] or multi-view (akin to digital 3D modeling with frequent view manipulation) eg. [Igarashi et al. 1999; Fan et al. 2004; Nealen et al. 2007; Bae et al. 2008; Fan et al. 2013]. Single-view use geometric properties of the 2D sketch to infer its depth in 3D, while multi-view techniques create the 3D curve explicitly using view manipulation to specify 3D curve attributes from different views. Our work lies in-between: we are able to infer 3D curves from strokes drawn

entirely in a single-view, but are less constrained to the view and global sketch context than other single-view approaches, allowing view manipulation as and when desired.

While several sketch-based interfaces have used template 3D geometry as a canvas on which to project 2D sketch strokes, for example [Kara and Shimada 2007; Nealen et al. 2007; Takayama et al. 2013], few techniques barring Overcoat [Schmid et al. 2011] have explored projecting strokes on and around underlying geometry. Focused on 3D painting, Overcoat provides some support for the creation of 3D curves normal to, or projected on, explicitly set offset surfaces of the underlying geometry. SecondSkin, in contrast, creates a network of curves, surfaces and volumes, by automatically classifying curve-types and inferring their depth on and around 3D geometry.

The abundance of planar curves and orthogonality in 3D sketches has been well exploited using cross-sections and principal direction curves [Schmidt et al. 2009; Andre and Saito 2011; Xu et al. 2014]. We capture these concepts using normal and tangent plane curve-types. contours, similarly have been extensively used in sketch-based modeling for: stroke inflation based modeling [Igarashi et al. 1999; Nealen et al. 2007; Olsen et al. 2011; Sýkora et al. 2014], multi-view modeling [Rivers et al. 2010] and model deformation [Zimmermann et al. 2007]. Perhaps, the closest use of contours to our work is in the context of sketching garments [Turquin et al. 2007; Robson et al. 2011], where a distance field from an underlying mannequin mesh in preset views is used to define a mesh, fit to contour and border strokes. For loose clothing, parts of the mesh distant from the mannequin conform to a revolved surface [Robson et al. 2011]. SecondSkin automatically classifies strokes perceived as lying on a shell of the underlying geometry, and creates network of 3D curves, surfaces and volumes depicting a layered 3D structure.

3 Understanding Layered Sketches

3D curve inference from 2D sketch strokes typically employs a curve fitness metric [Schmidt et al. 2009; Xu et al. 2014], with both discrete (for example to capture geometric regularity or curve-type constraints) and continuous terms (for example to model curve smoothness or minimal variation from the 2D stroke). This mix of prioritized discrete and continuous constraints makes fitness optimization for sketch understanding a challenging problem. Solutions such as combinatorial enumeration [Schmidt et al. 2009] or iterative least squares refinement [Xu et al. 2014] of discrete constraints, are tailored to exploit properties specific to the sketch problem domain. In our context of layered sketch strokes, prioritized discrete constraints take the form of admissible curve-types (Section 3.1), that we are able to efficiently classify from the 2D strokes and underlying 3D geometry. While we can similarly cast 3D curve inference as the optimization of a fitness function over a family of candidate 3D curves, our curve-type classification algorithm (Figure 4) allows us to find a suitably fit 3D curve directly.

3.1 Classifying 3D Curve-types

We formulated our 3D curve-type classification from discussions with artists, and observations of finished sketches. In sketches of layered models, such as Figures 2, and 10, a viewer's 3D interpretation of a sketch is strongly dependent on the perceived underlying geometry. Considering artists try to unambiguously define 3D form while sketching [Nakayama and Shimojo 1992], we assume that a smooth 2D stroke represents a similarly smooth 3D curve, and that inflections and points of high curvature often partition strokes into segments, where stroke sections can belong to a different 3D curve type. Our set of four 3D curve-types (Figure 3), are as follows:

Shell contour: curves are the contours of shells atop underlying

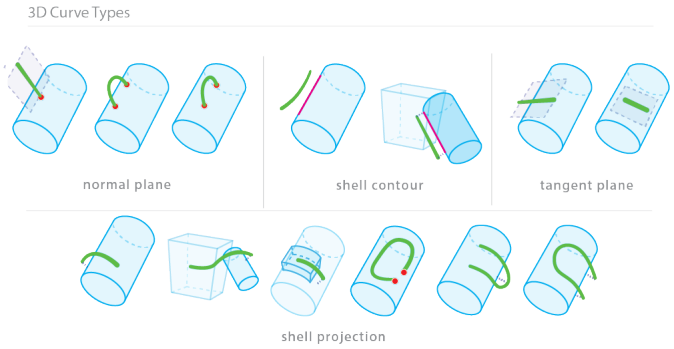


Figure 3: *Classification cases for 3D curve types.*

geometry. In general, shell is a loose term for a surface lying over another. Occluding contours and other feature curves of the underlying 3D geometry, that resemble the shape of a 2D stroke in the given view, however, are a strong perceptual cue of parallelism [Xu et al. 2014], and the perceived shell contour curve conforms in view depth to the corresponding contour of the underlying geometry.

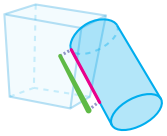
Shell Projection: curves similarly mark the boundary of a surface shell, but have no special view dependent meaning. In keeping with the minimal variation in view depth principle [Xu et al. 2014], we assume the height of a shell projection curve above underlying geometry, varies linearly along the curve between start and end height.

Tangent Plane: curves are planar variants of shell projection curves, in planes roughly aligned with the average surface normal of the underlying geometry. Strokes classified as shell projection curves can be alternately interpreted as tangent plane curves and vice versa.

Normal Plane: curves emanate from and roughly normal to the underlying surface. If both end points of a normal plane curve are attached to the surface, the curve lies along a plane defined by the surface attachments and their average surface normal.

Our classification algorithm is illustrated in Figure 4:

Shape similarity between a 2D stroke, and occluding contour or feature curve of the underlying geometry in the given view, is a perceptually strong cue, used first to classify strokes as shell contour curves. Occluding contours and other features like ridge and valley curves can be automatically extracted from underlying 3D geometry [Ohtake et al. 2004], or obtained naturally from the sketching process for previously sketched layers. We propose a small tolerance for shape deviation with little or no global rotation. Specifically, a stroke is tested against a segment of each occluding contour or pre-defined feature curve of the 3D geometry. The segment is defined between the closest points on the 2D contour to the stroke end-points, as shown inset where the green stroke matches a feature curve segment on the cylinder shown in purple.



If the midpoint distance, between stroke and curve segment, are less than δ_{mid} (δ defaults to $w/25$ in our implementation where w is the screen width), we transform (rigid with uniform scale) the stroke to align midpoints with the contour segment. We then compute a pairwise distance of (default $S/2$ where S is the stroke arclength) equispaced sample points along the two curves and accept shapes as matching only if the average point distance is not greater than δ_{avg} (default of $w/160$). If multiple feature curve segments are considered for matching we pick the segment with the lowest δ_{mid}. Strokes that match an occluding contour or feature curve segment are classified as shell contour curves.

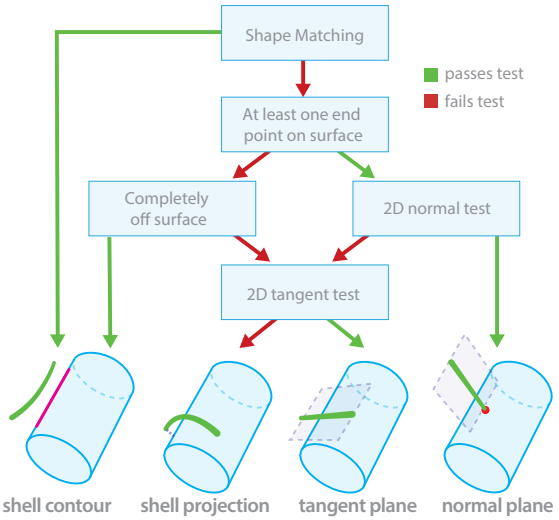


Figure 4: *Classifying strokes as one of 4 curve-types*

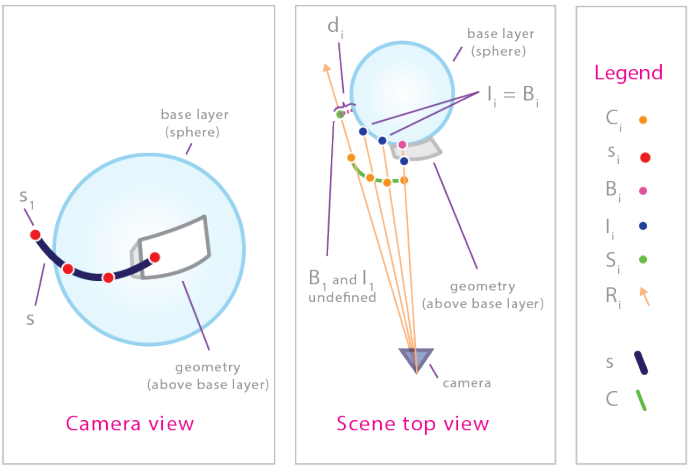
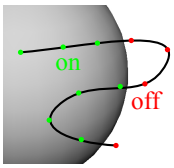


Figure 5: *3D curve inference terminology: a camera view where the user has drawn a stroke s containing points s_i (left); a top view showing a curve C with points C_i, as a possible inference solution for the stroke s (right). R_i is a ray from the camera through points s_i. The closest intersections of R_i with the base layer and with the closest geometric layer is B_i and I_i, respectively.*

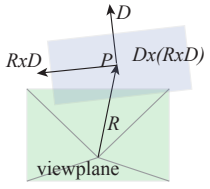
Figure 3 further, suggests a strong correlation between the 2D stroke and its end points relative to the underlying geometry. Let's say a 2D point is *on* or *off* an underlying surface if it projects or does not project onto the surface in the given view as shown inset. Strokes are completely *on* or *off* underlying geometry if *all* stroke points are on or off the underlying surface. All strokes that are not shell contour curves by virtue of shape matching, fall into one of 3 cases based on this property that help classify its curve type:



- stroke completely *off* underlying geometry: shell contour.
- both stroke end points are *off* underlying geometry but not the entire stroke: shell projection/tangent plane.
- at least one stroke end point is *on* underlying geometry: normal plane, shell projection/tangent plane.

Shell contours can thus be uniquely classified, but the remaining curve types require further processing. A normal plane curve, for instance requires that at least one end point is *on* underlying geometry. Normal plane curves are classified if and only if the stroke tangent t for all end points that lie on an underlying surface, aligns ($t \approx n$) with the view projected surface normal n at those end points. All remaining strokes are thus shell projection or tangent plane curves interchangeably. We distinguish shell projection and tangent plane curves based on the principle that the 2D stroke is indicative of its 3D shape [Xu et al. 2014]. If the stroke closely approximates a straight line, we thus imagine it as a 3D line lying in a tangent plane, else the stroke is a shell projection curve. We capture stroke straightness as the average of the Menger curvature at each stroke point, being less than a threshold ρ (default $\rho = 0.025$).

3.2 Inferring 3D Curves



Each curve-type further describes how a 3D curve may be inferred as a stroke of that type. A common 3D curve inference results in the stroke being projected in 3D onto a **minimum-skew viewplane**. Given a 3D point P and direction D, the minimum-skew viewplane is the plane best aligned with the viewplane containing P and D as shown in grey, inset. This plane is readily computed as $D \times (R \times D)$, where R is the ray from the eye to P.

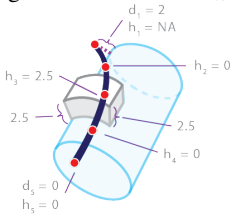
Inferring a 3D curve requires computing a 3D point C_i for all 2D stroke points s_i, $i \in 1, .., n$. The points C_i lie on the 3D view-ray R_i from the viewpoint through the 2D image point s_i. Curves are inferred with respect to a base layer, a user chosen geometric layer, and all layers below the base layer. Considering there can be layers above the base layer, let the nearest intersection between R_i and geometric layers, if defined be I_i and the nearest intersection between R_i and the base layer, if defined be B_i. When R_i does not intersect with geometry, let S_i denote the closest point on R_i to the base layer, with a distance d_i. Let h_i be a height function returning the distance between the point I_i and its closest point on the base layer. Figure 5 illustrates the above terminology.

We infer the different curve types in 3D as follows:

Shell contour: curves are created with one of two inference techniques. When shape matching, both the stroke and matching feature curve segment are resampled to have the same number of points. 3D points C_i are given by the closest points on rays R_i to corresponding 3D points on the feature curve segment.

In the absence of a matching contour shape, we create a minimum-skew viewplane lying on the contour of the base layer with $P = S_1$ and $D = S_n - S_1$, and 3D points C_i computed by projecting stroke points s_i onto this minimum-skew viewplane. As a result the 3D curve end points C_1 and C_n are S_1 and S_n.

Shell projection: If the curve end points are connected to known 3D positions, the height at each point on C_i is interpolated between the known heights of C_1 and C_n. Otherwise, we define a constant height value for the stroke as the maximum height $max(d_1, d_n, h_1, ..., h_n)$, based on the distance from the end points to the base layer contour, or the heights with respect to the base layer of projected stroke points I_i *on* higher geometric layers. For example, a stroke with associated h_i and d_i values, is shown inset, where the chosen constant height is given by $h_3 = 2.5$. Each point s_i is then projected on a distance offset surface of the



base layer with height given by the interpolated value at C_i. Stroke points that fail to project onto their corresponding offset surfaces, are projected to a minimum-skew viewplane between adjacent points that do project to an offset surface (the curve end points if required).

Normal Plane: curves require at least one end point *on* a surface. Say this point is s_0. We create a minimum-skew viewplane with $P = I_1$ and D as the surface normal at I_1. If both endpoints project to the surface, the point and direction at the two end points are averaged. 3D points C_i are the projections of stroke points s_i onto this minimum-skew viewplane.

Tangent Plane: curves are projections of the stroke onto a tangent plane whose normal is given by the average surface normals of all stroke points that project *on* the base layer. The height of the plane is determined similar to that of a shell projection curve (the average height is computed if the shell projection height varies along the curve). Tangent planes can pass through underlying geometry if the 3D position of both or one curve end-point(s) is already known. We revert in this case to a minimum-skew viewplane, defined between the two known end-points, or the one known end-point and the stroke point with the shallowest view-depth.

Finally, all 3D curves are smoothed along the rays R_i, by neighbour averaging their view depth.

While the above curve inference handles a majority of sketched curves, we provide some additional features for users to create strokes that do not fit our curve types and settle ambiguous situations. Occasionally a user might wish to construct a shell projection curve with a straight line, or conversely a tangent plane curve with considerable curvature. As the tangent plane curve-type is a planar variation of a shell projection, we allow the user to toggle our default interpretations using the **shift** key. More precisely, if the algorithm were to choose a shell projection curve and the **shift** key was pressed, the result would be swapped with a tangent plane curve, and vice-versa. Users can explicitly indicate curve types by mode switching but our strength lies in presenting a pure sketching interface where our algorithm enables a creative workflow that is not impeded by consciously mode-switching prior to placing a stroke.

3.3 3D curve quality and default planar curves

Once the 3D curve for a sketched stroke has been inferred, we evaluate its fitness quality. Various quality metrics such as smoothness, minimal shape variation, low foreshortening [Xu et al. 2014] and overall curvature [Schmidt et al. 2009] have been used in sketch literature. In general, fitness criteria are best adapted to the assumptions of the 3D inference technique. True2Form [2014] for example, implicitly captures smoothness by representing curve segments using cubic beziers, and foreshortening modeled as the overall variation in view depth of adjacent Bezier control points.

In the context of layered sketches, where the view depth of stroke points are computed independent of its neighbouring points, we find curve smoothness to be an important fitness criterion. Considering curves are segmented by points of high curvature we expect resulting curves to be reasonably smooth. Therefore, if the max variation in the view depth of the generated curve is greater than ν (default ν is 0.4) the curve is deemed not smooth. As surface normals aligned with the view direction can result in highly foreshortened normal plane curve projections, low foreshortening is another fitness criterion. If v_{min}/v_{max} (min over max view depth of the curve) is lower than μ (default μ is 0.5), the curve is considered overly foreshortened. In practice this happens rarely, but in the case of these classified 3D curves of poor quality or *No-Type* curves shown in the user study of Section 5, we replace the interpreted curve by the fittest minimum skew viewplane or XYZ planar curve. In other words, if just one of the curve end-points is defined in 3D we project the stroke onto the least foresfhortened of three planes aligned with the XYZ axes and passing through the 3D end-point. If both end-points are known in 3D we project the stroke onto the minimum-skew viewplane passing through the two 3D end-points. These alternate planar curve interpretations can also be forced by a user holding the **space** key.

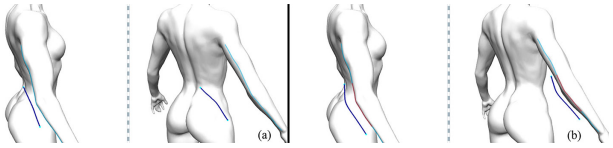


Figure 6: *Shell contour curve with shape matching: image pairs show a sketched stroke (blue) on the left and the result from a different view on the right. A shape match between the stroke and a feature curve (teal) is attempted. No match results in a shell projection curve, image-pair (a), and a matched sub-curve (red), results in a shell contour curve, image-pair (b).*

4 Implementation

SecondSkin is an interactive 3D sketched-based modeling system, that facilitates single-view sketching but also allows the user to manipulate the camera freely. The interface is designed to be used with a graphics tablet. We implemented our system on a PC with an i7 Intel CPU, 8 GB of RAM, and an ATI Radeon HD 5700. The system runs completely in real-time with a barely noticeable pause when interactively constructing mesh surfaces and volumes. The system uses a kd-tree for 3D distance calculations used to to determine height values between newly sketched strokes and the base layer. Distance offset surfaces are approximated on the GPU using a vertex shader to inflate the mesh. We find this method reasonable, since these surfaces are only used for stroke projection from the current view.

4.1 Layered Curve, Surface and Volume Workflow

We provide typical sketch based modeling functions such as erase gestures, stroke smoothing, and global symmetry. Users sketch strokes, like with pen and paper (see accompanying video), to construct curves, surfaces, and volumes in the scene. While we typically expect smooth strokes, we do segment closed loops to form surfaces and volumes, at inflections and points of high curvature. Once a segment in the curve is detected a 2D preview of the completed closed loop is shown to the user. At any time there is an active base layer that forms the geometry with respect to which new strokes are inferred in 3D and placed in the active drawing layer. All layers below the active drawing layer form the underlying 3D geometry. Geometric layers above the active drawing layer, if present, are passive and are simply a visual reference. Although SecondSkin has limited layer editing and management functionality, a layer editor similar to that found in image editing software like Adobe Photoshop is easy to envisage and implement. Figure 7 shows a hypothetical user interface for such a layer editor.

Our system has 3 geometric primitives: curves, surfaces, and volumes. These primitives are defined using a solid modeling structure, where surfaces are bound by a closed set of curves, and volumes bound by a closed set of surfaces. The system uses a graph structure to maintain connections between primitives. When closed loops are detected in the curve graph a surface is constructed. A user can also form a closed loop by sketching a smooth stroke connecting 3D curves in the scene. The user can also construct a closed loop from

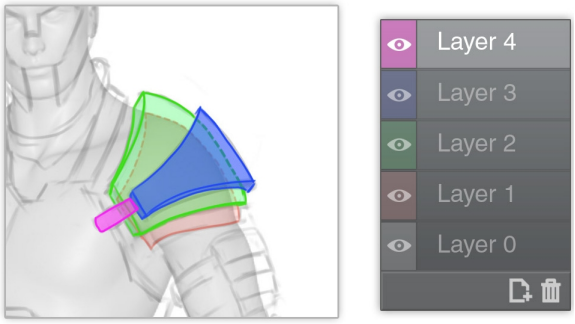


Figure 7: *A hypothetical user interface for layer editing functionality commonly found in image editing software.*

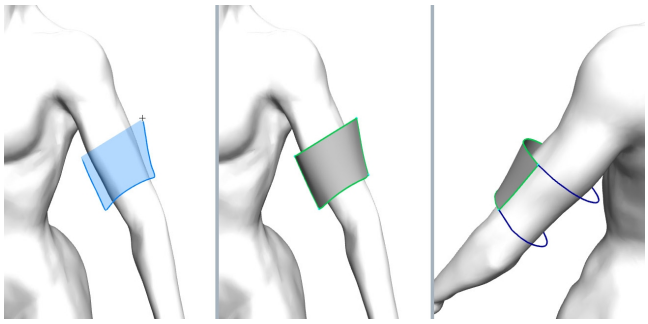


Figure 8: *Surface envelopment example where a user creates an armband. The image panels from left to right show: the closed loop sketch; the result from the sketching view; and, the result from another angle showing the back of the armband.*

a single view, and in this case the system assumes the surface will enclose a volume.

4.1.1 Surface Creation From 2D Closed Loop

Users can draw surfaces and indeed volumes using a closed curve loop from a single-viewpoint. The curve loop is entirely composed of shell contours and shell projections or tangent plane curves. Whenever possible, these strokes snap at their end-points to anchors (existing 3D curve end points, or edges). The strokes, if classified as shell projections or tangent plane curves, then inherit their height values from these anchor points. In the absence of adjacent anchors, the curve graph structure may have to be traversed to compute appropriate height values for such curves. In this case a breadth first search is performed outward from an end point of a shell projection curve until 3D anchor points are reached. The height value weight of such an anchor is inversely proportional to the number of edges traversed to reach it in the graph and the estimated height value is simply a weighted average of these connected height values.

Once the 3D curves in the loop have been inferred, surfaces are constructed with 3 and 4 sided Coons patches. Closed loop curves are combined and split to form a set of 3 or 4 curves using a simple meshing strategy (Section 4.2).

It is common in layered sketching to produce shell volumes that wrap around underlying geometry. Shell volumes like the armband in Figure 8 can be created by joining two half-shells drawn from opposite points of view. We provide automated support however, for this common structure: closed loops that contain 2 non-adjacent shell contour curves are assumed to envelope the underlying geometry. A *back* loop is created to match the sketched loop in *front*. As seen in Figure 8, the four end-points of the inferred 3D shell contour curves define an average plane of reflection. The remaining curves in the *front* loop are reflected about this plane, offset the same distance from the back-facing geometry as its front-facing counterpart, and smoothly connected to the front loop to define a *back* loop.

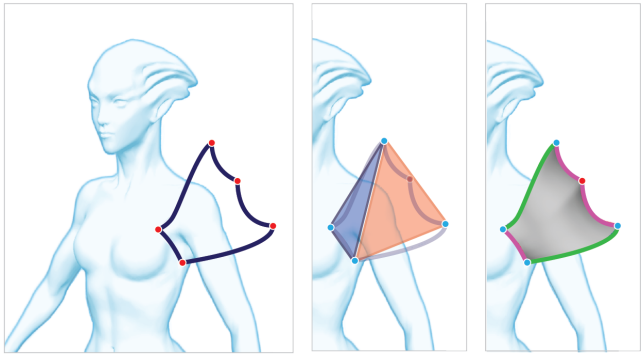


Figure 9: *An example of our Coons patch configuration when more than 4 curves are present. In this case, 5 curves are combined into a 4-sided Coons patch. Left: the original closed loop of 5 curves. Middle: choice of 4 vertices that produces the maximum footprint quadrilateral (maximum of triangle areas of the two possible triangulations). Right: the final Coons patch showing the chosen curve combination.*

4.1.2 Volume Creation

The surface constructed when a user draws a closed loop in a single view is assumed to implicitly enclose a volume with the base layer. We construct a volume automatically by constructing new surfaces between the base layer and newly constructed surface. This type of volume can also be constructed with any surface by clicking on the surface while holding the **ctrl** key. An enclosed volume is not always desired, so we provide an extrusion technique for creating volumes as well. While holding down the **ctrl** key, the user sketches a normal curve starting on the surface acting as the base of the volume. The volume is then extruded along the normal curve.

4.2 Meshing

Given a closed loop, the meshing system finds a suitable set of curves for Coons patching. Coons patching requires 3 or 4 curves, so we split curves in closed loops of 2 or less and we combine curves in closed loops of 5 or more. When splitting curves, we split the curves into regions of equal arclength until we have 4 curves. Given 5 or more curves we must combine curves until 4 curves are left. Combining curves is less trivial as the choice of curves affects the final look of the mesh. We use a simple strategy that looks at every C_4^n combination of 4 vertices of an $n - sided$ curve loop. We simply pick the configuration with the maximum footprint quadrilateral (maximum of triangle areas of the two possible triangulations) connecting the 4 points. These form the corners of a 4-sided Coons patch. An example of a curve combination is shown in Figure 9 where 5 curves are combined into a 4-sided Coons patch. While this works reasonably in practice, an approach based on design driven quadrangulation [2012] would be more general and robust. In addition, the quad meshing approach of Yassen et al. [2013], specifically designed for garment modeling, would provide better results when meshing cloth features.

5 Perceptual Validation

We performed a study to formally test consistency between humans and our algorithm, on classifying sketch strokes in layered structure sketches. We aim to answer two questions:
Q1: Do humans consistently perceive 2D sketch strokes layered over underlying geometry?
Q2: Does human perception, when consistent, match our algorithmic output for the above strokes?

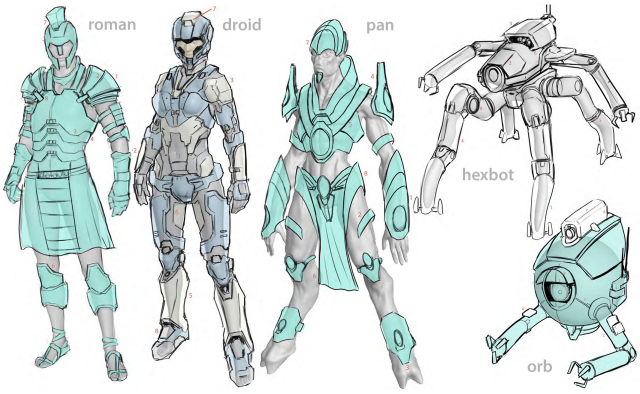


Figure 10: *User study corpus comprising 5 sketches.*

Study Design: Our test data-set comprised 5 drawings, created by us for uniformity, but carefully done to match the spirit of drawings by the artist community (Figure 10). The drawings comprise a mix of human and robotic forms, typical of layered 2D concept art (Figure 2): *droid*, *pan* and *orb* are inspired by Microsoft Studios' Halo, Blizzard Entertainment's Starcraft, Valve Corporation's Portal, series of games respectively. We formulated our study as a set of 40 curve-type queries (8 per sketch). The number of sketches and questions was chosen so the typical time to complete the study was ≈ 10 mins. The four curve-types were verbally explained to participants using illustrations (this was done on drawings that were not used as testing data). Users were then asked to imagine each sketch as a layered 3D model, possibly with multiple layers like *orb* in Figure 10 and then classify 8 query curves as one of the four curve types or no-type (to reduce any bias towards the set of curve types). Queried curves were unobtrusively numbered avoiding perceptual bias possible when using color or visual markings. The study was performed by 15 participants on print-outs, 10 of whom had some computer graphics background.

Study Results:
Q1: For the 40 queried curves the histogram of agreement on the dominantly perceived curve type was $[90-100\% : 20, 70-90\% : 11, 50-70\% : 4, 40-50\% : 3]$. 15 curve queries had 100% agreement and only 3 fell outside the strong significance threshold of $p = 0.01$. In other words, only 3 curves could statistically belong to any of the 5 curve types. One such blue curve can be seen near *roman's* knee in Figure 11 that 40% of the users imagined as a tangent plane curve but clearly has alternate interpretations. The dominantly perceived curve types for the 40 queries were spread over [contour: 13, projection: 9, tangent: 7, normal: 8, no-type: 3]. Normal plane curves when dominantly perceived, always had high consistency of 87% or more. As expected, in the case of 7 curves some perceptual ambiguity was observed between shell projections and tangent plane curves, such as the blue curve on top of *orb* in Figure 11, where viewers were split 47%-53% between a shell projection and tangent plane.

Q2: Our algorithm agreed with the majority of participants on all but 1 out of 40 curve queries. The anomalous curve was one of the 7

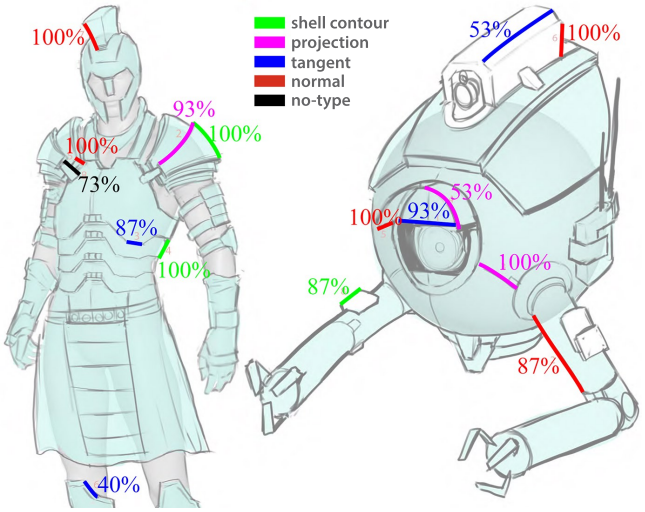


Figure 11: *Queries and results (%age human consistency for the dominant response) illustrated on 2 models.*

curves that could be classified as tangent plane or shell projection. In these cases, our algorithm uses the shape of the 2D stroke to distinguish the two types. A near straight line stroke in keeping with the non-accidental viewpoint assumption is classified as tangent plane and others as shell projections. Designers can readily create the alternate interpretation in our system using the **shift** key. In many cases the geometric difference in the resulting 3D curve between these two interpretations is also subtle.

The 3 curves dominantly viewed as no-type, such as the black curve connecting *roman's* shoulder and chest armour in Figure 11, are adjacent to strokes of known curve-type. Our algorithm by default reconstructs these strokes as shell projections once their end-points have been defined in 3D by adjacent curves. Exhaustive classification of all curves on *roman* by one viewer in Figure 1(b) reported only 9% strokes as having no-type, and these were conveniently reconstructed in 3D in context of their adjacent strokes. In summary, we believe this study provides sufficient validation that viewers consistently perceive sketch strokes in layered 3D models, and our algorithm is able to reliably match viewer expectation.

6 Conclusion

Evaluation: A set of 6 users tried our system, one artist and 5 non-artists, all with some CG background. The non-artists were given the underlying 3D model and produced the results shown in Figure 12, within a 1.5 hour modeling session that included learning the controls of the system and familiarizing themselves with the workflow. Figure 13 is the work of the artist. The underying geometry for the bike and orb robot was a sphere, and the spider tank began with a blobby model of similar form. All models in Figures 12, 13 took 20-40 mins, except the top left model in Figure 13 (13 min). Producing results of this quality and efficiency using existing techniques would be quite cumbersome. The users in general found the system fun and creative to use. Creating man-made models was more difficult than armour on characters but the addition of simple primitive modeling would greatly improve our industrial design modeling.

We observed that the artist was comfortable sketching outer most layers early on in the modeling process, where as the non-artists always started with the lowest layer. One of the non-artists mentioned that he felt a need to create the lowest layer first before moving higher. Conceptually this seems logical as underlying layers are expected to be enclosed by higher layers. The artist noted that he would instead

Figure 12: *SecondSkin creations by a group of non-artists*

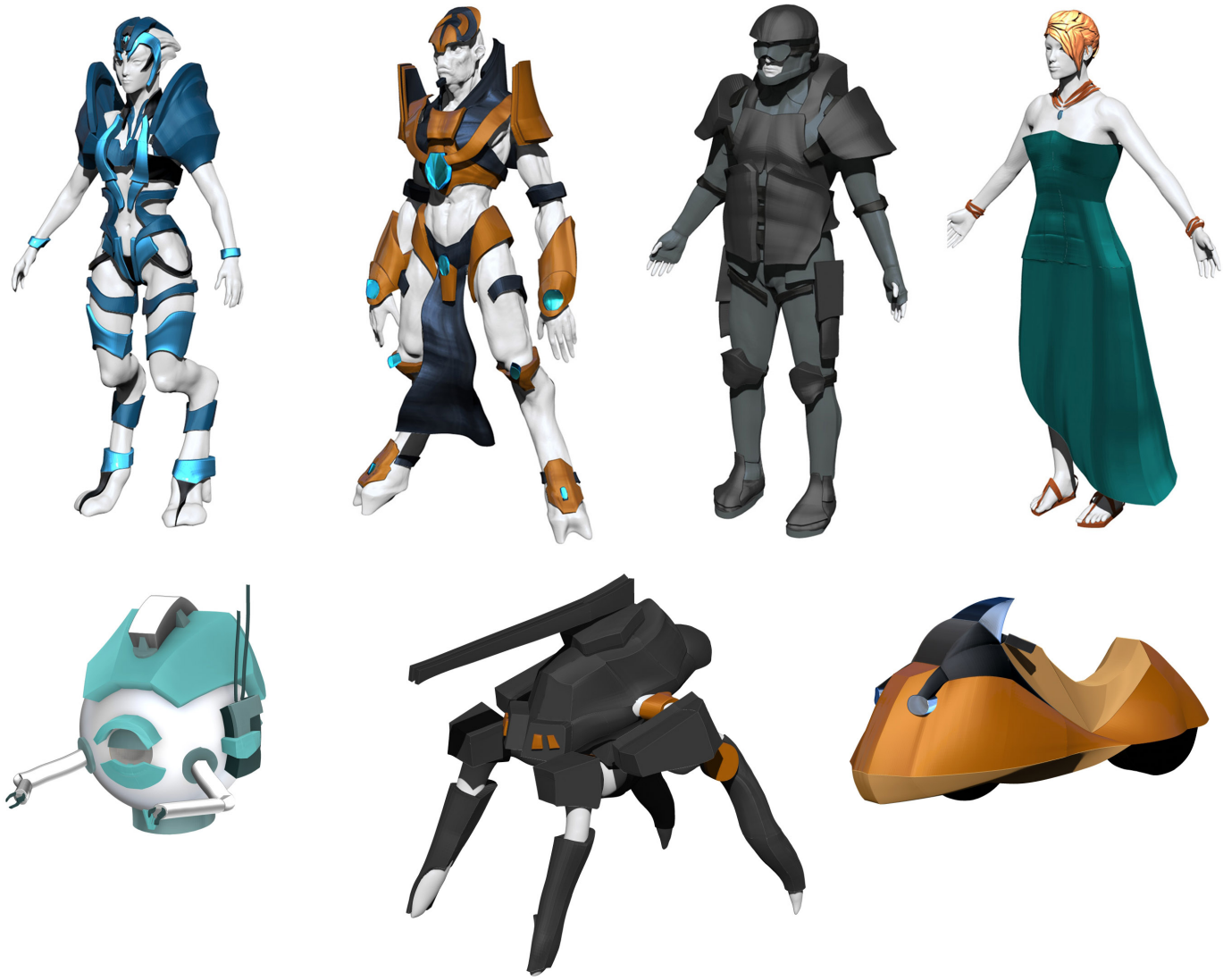


Figure 13: *Artist creations using SecondSkin*

	Alien	%	Sword	%	Soldier	%
Time	16:48		12:35		19:44	
Curves created	166	-	86	-	219	-
Curves in the model	129	77.7	65	75.6	160	73.1
Closed loop and volume curves	86	51.8	27	31.4	123	56.2
Misclassified curves	5	3.0	7	8.1	14	6.4
Forced planar curves	18	10.8	34	39.5	3	1.4
Swapped curves	2	1.2	2	2.8	1	0.5

Figure 14: *Usage statistics captured during three modeling sessions by the artist. Models are shown on the right. Where applicable, percentages are shown with respect to the number of curves created.*

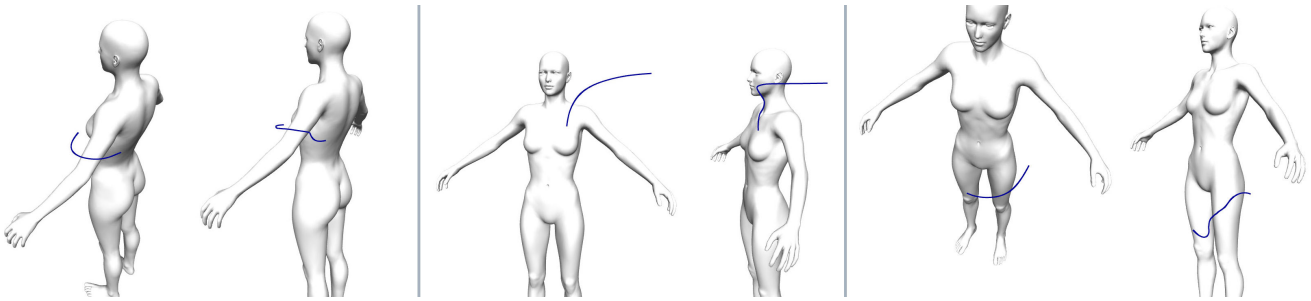


Figure 15: *Three examples of the most common classification error where strokes which appear to be planar curves lying on one of the cardinal X, Y, Z planes are classified as shell projections. Each pair of images shows the sketching view on the left and a second view showing the result on the right. These can be corrected with a modifier key to enforce the desired curve type.*

sketch in large forms as he would in traditional sketching, producing structures that would be found on the outer most layers. The most problematic feature for all users was determining and controlling the height of a normal plane curve's end point due to typically large foreshortening. Providing post-sketch editing control over the height values and indeed all parameters of the solid 3D models is subject to future work.

We present a set of usage statistics in Figure 14, captured over three modeling sessions by the artist. For these sessions, we added a second form of delete to explicitly indicate that a curve was misclassified. The number of misclassified curves averaged 5.8% over the three models. In the sword model, 39.5% curves were forced planar, typical for industrial design objects and far more than used in organic forms. We can also note that the artist predominantly used the closed loop and volume tools to create the majority of curves.

Limitations: Figure 15 shows three examples of poorly classified and inferred 3D curves. Part occlusions in the interior of the underlying geometry can result in poor 3D curve inference. Strong foreshortening and ambiguous viewpoints can also produce undesirable results. We address these both by defaulting poor quality 3D curves to appropriate planar curves. In the case of Figure 15, our fitness tests for poor quality do not catch the resulting errors. Exploring further fitness criteria is the subject of future work. Our system also has limited support for curve, surface and volume editing as users typically erase and redraw undesirable curves. Our choice of Coons patches also limits continuity between neighbouring patches to C_1 continuity when possible.

Discussion: Shell-like layered structures are increasingly relevant with the onset of multi-material 3D printing technology. While game characters were our inspiration, our system is capable of modeling both man-made and organic forms (Figure 13). We can draw almost any curve network by using many normal curves as scaffolding on which to anchor arbitrary 3D curves, but that is not what makes SecondSkin compelling. We are capable of but not streamlined to create curves with no connection to the underlying geometry, conically shaped shells, holes, and bumps. Note, that SecondSkin can be easily augmented by various known curve regularization approaches, and widgets, for example to scale our default cylindrical volumes into conical shells. We eschew these to avoid confusion between prior art and our novel contributions, and leave their integration into SecondSkin as future work.

In summary, we have presented SecondSkin, the first system aimed at sketch-based modeling of layered 3D structures. We believe the worlds of conceptual design and downstream solid modeling will merge in the future. Our work is a promising step towards that goal.

References

ANDRE, A., AND SAITO, S. 2011. Single-view sketch based modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, ACM, New York, NY, USA, SBIM '11, 133–140.

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '08, 151–160.

BESSMELTSEV, M., WANG, C., SHEFFER, A., AND SINGH, K. 2012. Design-driven quadrangulation of closed 3d curves. *ACM Trans. Graph. 31*, 6 (Nov.), 178:1–178:11.

BLOOMENTHAL, J., AND WYVILL, B., Eds. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

CHADWICK, J. E., HAUMANN, D. R., AND PARENT, R. E. 1989. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 243–252.

CHEN, T., ZHU, Z., SHAMIR, A., HU, S.-M., AND COHEN-OR, D. 2013. 3sweepp: Extracting editable objects from a single photo. *ACM Trans. Graph. 32*, 6 (Nov.), 195:1–195:10.

CUTLER, B., DORSEY, J., MCMILLAN, L., MÜLLER, M., AND JAGNOW, R. 2002. A procedural approach to authoring solid models. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '02, 302–311.

DAVIES, P., AND SCOTT, K. 2012. *Awakening: The Art of Halo 4*. Titan Books Limited.

FAN, Z., CHI, M., KAUFMAN, A., AND OLIVEIRA, M. M. 2004. A Sketch-Based Interface for Collaborative Design . In *Sketch Based Interfaces and Modeling*, The Eurographics Association, J. A. P. Jorge, E. Galin, and J. F. Hughes, Eds.

FAN, L., WANG, R., XU, L., DENG, J., AND LIU, L. 2013. Modeling by drawing with shadow guidance. *Computer Graphics Forum (Proc. Pacific Graphics) 23*, 7, 157–166.

IGARASHI, T., AND MITANI, J. 2010. Apparent layer operations for the manipulation of deformable objects. In *ACM SIGGRAPH 2010 Papers*, ACM, New York, NY, USA, SIGGRAPH '10, 110:1–110:7.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 409–416.

KARA, L. B., AND SHIMADA, K. 2007. Sketch-based 3d-shape creation for industrial styling design. *IEEE Comput. Graph. Appl. 27*, 1 (Jan.), 60–71.

NAKAYAMA, K., AND SHIMOJO, S. 1992. Experiencing and Perceiving Visual Surfaces. *Science 257*, 1357–1363.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh: designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, SIGGRAPH '07.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 609–612.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2009. Sketch-based modeling: A survey. *Computers and Graphics 33*, 1, 85 – 103.

OLSEN, L., SAMAVATI, F., AND JORGE, J. 2011. NaturaSketch: Modeling from images and natural sketches. *IEEE Computer Graphics and Applications 31*, 6, 24–34.

PRATSCHER, M., COLEMAN, P., LASZLO, J., AND SINGH, K. 2005. Outside-in anatomy based character rigging. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '05, 329–338.

RIVERS, A., DURAND, F., AND IGARASHI, T. 2010. 3d modeling with silhouettes. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 109:1–109:8.

ROBSON, C., MAHARIK, R., SHEFFER, A., AND CARR, N. 2011. Context-aware garment modeling from sketches. *Computers and Graphics (Proc. SMI 2011)*, 604–613.

SCHMID, J., SENN, M. S., GROSS, M., AND SUMNER, R. W. 2011. Overcoat: An implicit canvas for 3d painting. In *ACM SIGGRAPH 2011 Papers*, ACM, New York, NY, USA, SIGGRAPH '11, 28:1–28:10.

SCHMIDT, R., AND SINGH, K. 2008. Sketch-based procedural surface modeling and compositing using surface trees. *Comput. Graph. Forum 27*, 2, 321–330.

SCHMIDT, R., AND SINGH, K. 2010. Meshmixer: An interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*, ACM, New York, NY, USA, SIGGRAPH '10, 6:1–6:1.

SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3d scaffolds. *ACM Trans. Graph. 28*.

SHAPIRO, V., FARIN, G., HOSCHEK, J., AND S. KIM, M., 2001. Solid modeling.

SUTHERLAND, I. E. 1964. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, ACM, New York, NY, USA, DAC '64, 6.329–6.346.

SÝKORA, D., KAVAN, L., ČADÍK, M., JAMRIŠKA, O., JACOBSON, A., WHITED, B., SIMMONS, M., AND SORKINE-HORNUNG, O. 2014. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transaction on Graphics 33*.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4 (July), 97:1–97:8.

TURQUIN, E., WITHER, J., BOISSIEUX, L., CANI, M.-P., AND HUGHES, J. F. 2007. A sketch-based interface for clothing virtual characters. *IEEE Comput. Graph. Appl. 27*, 1 (Jan.), 72–81.

VAILLANT, R., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July), 125:1–125:12.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2000. *Virtual Clothing: Theory and Practice*. No. v. 1 in Virtual Clothing: Theory and Practice. Springer.

XU, B., CHANG, W., SHEFFER, A., BOUSSEAU, A., MCCRAE, J., AND SINGH, K. 2014. True2form: 3d curve networks from 2d sketches via selective regularization. *Transactions on Graphics (Proc. SIGGRAPH 2014) 33*, 4.

YASSEEN, Z., NASRI, A., BOUKARAM, W., VOLINO, P., MAGNENAT-THALMANN, N., ET AL. 2013. Sketch-based garment design with quad meshes. *Computer-Aided Design*.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2007. Silsketch: Automated sketch-based editing of surface meshes. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, ACM, New York, NY, USA, SBIM '07, 23–30.