Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling

CHANGJIAN LI*, The University of Hong Kong and Microsoft Research Asia
HAO PAN, YANG LIU, and XIN TONG, Microsoft Research Asia
ALLA SHEFFER, University of British Columbia
WENPING WANG, The University of Hong Kong

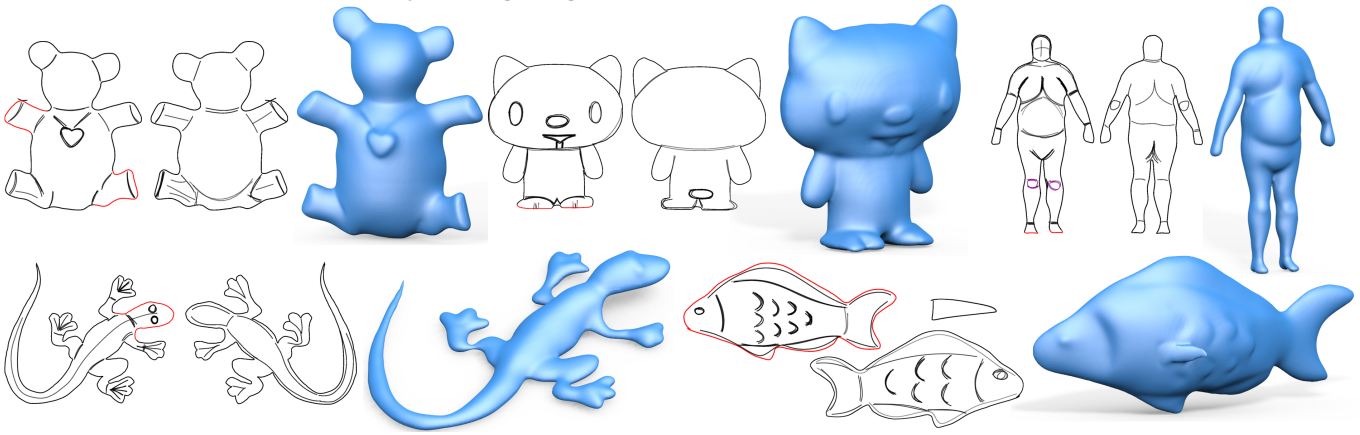


Fig. 1. Sketches and the corresponding 3D shapes computed by our method. Portions of the contour curves (red) are re-sketched to provide nontrivial boundary position data (Sec. 4), lifting teddy limbs, gecko head, and bending fish body. Purple curves of the human sketch are specified with target curvature hints (Sec. 3.1). For each sketch, our method uses a CNN to predict the 3D surface patch. Two or more surface patches are fused into the complete shapes. Back (bottom) views reuse contour curves of the front (top) views; only interior strokes are drawn there. The triangle sketch for the fish is used to model its side fins. Note that darker and over-sketched strokes generally correspond to stronger curvatures, which looks natural and intuitive.

Sketching provides an intuitive user interface for communicating free form shapes. While human observers can easily envision the shapes they intend to communicate, replicating this process algorithmically requires resolving numerous ambiguities. Existing sketch-based modeling methods resolve these ambiguities by either relying on expensive user annotations or by restricting the modeled shapes to specific narrow categories. We present an approach for modeling generic freeform 3D surfaces from sparse, expressive 2D sketches that overcomes both limitations by incorporating convolution neural networks (CNN) into the sketch processing workflow.

Given a 2D sketch of a 3D surface, we use CNNs to infer the depth and normal maps representing the surface. To combat ambiguity we introduce an intermediate CNN layer that models the dense curvature direction, or flow, field of the surface, and produce an additional output confidence map along with depth and normal. The flow field guides our subsequent surface reconstruction for improved regularity; the confidence map trained unsupervised measures ambiguity and provides a robust estimator for data fitting. To reduce ambiguities in input sketches users can refine their input by providing optional depth values at sparse points and curvature hints for strokes. Our CNN is trained on a large dataset generated by rendering sketches of various 3D shapes using non-photo-realistic line rendering (NPR) method that mimics human sketching of free-form shapes. We use the CNN model to process both single- and multi-view sketches. Using our multi-view framework users progressively complete the shape by sketching in different views, generating complete closed shapes. For each new view, the modeling is assisted by partial sketches and depth cues provided by surfaces generated in earlier views. The partial surfaces are fused into a complete shape using predicted confidence levels as weights.

We validate our approach, compare it with previous methods and alternative structures, and evaluate its performance with various modeling tasks. The results demonstrate our method is a new approach for efficiently modeling freeform shapes with succinct but expressive 2D sketches.

CCS Concepts: • **Computing methodologies** → **Shape modeling**;

Additional Key Words and Phrases: Freeform surface, sketch, convolutional neural network, direction field, robust statistics, multiple view

ACM Reference Format:
Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2018. Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling. *ACM Trans. Graph.* 37, 6, Article 238 (November 2018), 12 pages. https://doi.org/10.1145/3272127.3275051

*This work was done when Changjian Li was an intern at Microsoft Research Asia.

0730-0301/2018/11-ART238 $15.00
https://doi.org/10.1145/3272127.3275051

1 INTRODUCTION

2D line sketch depicts rich geometric features of 3D shapes such as silhouettes, occluding and suggestive contours, ridges and valleys and hatching lines, and thus provides a succinct and intuitive way for 3D shape illustration, for which a set of non-photo-realistic rendering (NPR) methods [Cole et al. 2008, 2009; DeCarlo et al. 2003;

Hertzmann and Zorin 2000; Judd et al. 2007] have been successfully developed to automatically extract expressive sketches from 3D shapes. The inverse process of modeling 3D shapes from sparse 2D sketches also demonstrates its great potential for modeling free-form surfaces that are difficult for traditional CAD systems. However, this inverse problem is much more challenging because of the inherent ambiguity due to sketch simplicity and 2D to 3D dimension elevation. Many previous works apply hand crafted geometric priors to remove the ambiguity; examples include [Igarashi et al. 1999; Nealen et al. 2007] for modeling smooth surfaces of the biharmonic type which fit to the lines, [Chen et al. 2013; Shtof et al. 2013] for symmetrical generalized cylinders and other primitives which have the lines as profiles, [Iarussi et al. 2015; Li et al. 2017; Xu et al. 2014] for freeform shapes with principal curvature lines traced by the sketched lines, and [Jung et al. 2015] for developable surfaces with lines depicting folds, etc. While these methods have great modeling capability, they generally require the user to annotate the sketched lines in detail so that their respective geometric priors can be applied. Recently, emerging works use data-driven approaches and train an all-encompassing machine learning model, e.g. a convolutional neural network (CNN), to map the input sketches to 3D shapes directly [Delanoy et al. 2017; Lun et al. 2017]. These methods resolve the ambiguity issue by building a specific machine learning model for each object category (e.g. chairs, planes, characters), which however restricts the modeled shapes to the categories having large 3D datasets for training.

In this paper, we present a sketch-based approach that enables modeling generic freeform surfaces and complete 3D shapes with sparse but expressive user interactions. In particular, the user draw line sketches in a 2D plane to model a surface patch represented by depth and normal map; sketching incrementally in multiple viewing planes produces surface patches that form complete shapes. The core of our method is a CNN that maps input 2D line sketches to freeform surface patches that are naturally regular and capture the intention of sketch.

We train the CNN with a large generated dataset that has diverse 3D shapes and planar sketches rendered by NPR methods that mimic how humans depict the models. A naive CNN structure trained for such a task performs poorly, due to the inherent ambiguity of varying 3D shapes corresponding to the same sparse input sketches. Without category-specific priors for modeling generic freeform surfaces, we resolve ambiguity by using basic geometric and statistical principles. First, instead of mapping from sketch to geometry directly, we introduce an intermediate layer of dense curvature direction (flow) field, which the CNN first regresses and then combines with the sketch to infer the result surface. Flow field provides dense local guidance for geometry inference and leads to more regular surfaces with enhanced variation responding to sketched lines. Second, we explicitly model the perceived ambiguity of a sketch with a confidence map that is low on uncertain regions and high otherwise. While apparent to human observers, the ambiguity or confidence map is hard to quantify with hand-crafted rules. We let CNN predicts the confidence map instead, and train it in an unsupervised way through the use of robust statistics. Finally, we allow the user to modify the predicted surfaces, by making the CNN capable of taking optional user specifications of sparse points with depth cues, or sharp features and surface curvature hints along strokes.

Based on single view surface modeling, we also develop a multi-view system to progressively create a complete 3D shape by sketching in different views. After a surface patch is sketched in a previous view, the user rotates and updates the partial 3D shape by sketching in the new view, where partial sketches rendered and depth cues sampled from existing surfaces can assist the modeling. Finally, a complete 3D shape is generated by fusing the surfaces weighted by their corresponding confidence maps.

We validate our approach with extensive examples, and compare it with previous methods, alternative structures, and also other networks for modeling category-specific objects. Results show our network produces regular shapes with rich variations and subtle details while requiring less user input. Evaluation by novice users also shows that our tool makes the modeling of freeform shapes efficient and accessible for common users.

2 RELATED WORK

Sketch-based freeform shape modeling has accumulated a rich literature throughout the decades of research. However, we very roughly divide the previous works into two groups, one being the more classic geometric inference methods that rely on geometric priors to reconstruct 3D shapes, and the other being data-driven methods that build machine learning models to learn priors from particular 3D shape datasets and infer shapes. Our approach actually borrows ideas from both groups, as we foremost build on geometric principles, including the mapping between projected curvature directions and 3D shapes and normal/depth consistency, to solve the generic freeform modeling problem, but also draw on the powerful fitting capability of CNNs to carry out the many-factor nonlinear tasks of parsing sketches and generating dense curvature direction fields, and mapping from the projected geometric data to spatial depth and normal. Below we briefly discuss the most related previous works.

Geometric inference methods. A large number of previous works have focused on developing effective geometric principles, based on which it is possible to accurately infer 3D shapes from user sketches without ambiguity. The user sketched contours combined with membrane functionals [Igarashi et al. 1999; Joshi and Carr 2008; Nealen et al. 2007; Yeh et al. 2016; Zhang et al. 2001] or other smooth interpolation functions [Bernhardt et al. 2008; Olsen et al. 2011; Schmidt et al. 2005] can quickly determine smooth low-frequency 3D shapes. Sketched profile curves deform and blend user annotated primitives (e.g. generalized cylinders, pyramids, etc) to form complex shapes [Chen et al. 2013; Gingold et al. 2009; Shtof et al. 2013]. Based on how artists depict freeform shapes with a regulated combination of contours, features, and representative surface curvature lines, a series of works [Bae et al. 2008; Iarussi et al. 2015; Li et al. 2017; Schmidt et al. 2009; Shao et al. 2012; Xu et al. 2014] resolve the 2D to 3D ambiguity and convert planar sketches to 3D data.

Our method uses a data-driven approach to learn the geometric inferences based on the same set of principles, including shape regularity and shape from contour, feature and representative curvature lines. On the other hand, while these previous methods rely on detailed user annotations to parse the sketches into curves of

different functions and often use expensive nonlinear numerical optimizations to solve the 2D to 3D conversion, we utilize CNN models to parse the sketch and infer the geometry with improved efficiency and reduced user sketch and annotations. See Sec. 5.2 for comparisons.

Data-driven methods. For many common objects and scenes, it is usually reasoned that we humans envision their 3D shapes by first recognizing what they are and then matching a prior shapes of the same category in memory to the observations. This idea underlies a range of data-driven methods for sketch-based modeling, as they generally separate the modeling task into two steps: first search matching shapes through a database against an input sketch, and then adapt the retrieved shapes as necessary to fit the input sketch. Examples include pure sketch-based retrieval [Eitz et al. 2012; Su et al. 2015; Wang et al. 2015c], and retrieval with subsequent adaptation and composition, like Sketch2Scene [Xu et al. 2013] for scene modeling and [Guo et al. 2016; Lee and Funkhouser 2008; Xie et al. 2013] for object modeling. While these approaches significantly ease the user burden by providing abundant a prior knowledge for a specific category of objects that the user tries to model, the tool built for one category however does not generalize to others. In comparison, our machine learning model learns the more generic geometric reconstruction process rather than the knowledge of class-specific 3D objects, which makes our method possibly less efficient for modeling a particular class of objects but more generic with finer levels of shape control provided to the user.

Later works in this domain do not explicitly separate the model searching and adapting steps, but rely on the powerful deep neural networks to map directly from sketch to 3D data, examples including [Delanoy et al. 2017; Lun et al. 2017; Su et al. 2018]. [Su et al. 2018] predicts normal maps from a category-specific 2D sketch by an encoder-decoder network, which minimizes normal fitting loss and adversarial loss, and takes as optional input user specified normal samples. [Delanoy et al. 2017] uses a CNN to map sketches to a volumetric occupancy grid representing the 3D shape, and allows the incremental update of the shape through an updater CNN as the user sketches in new views. However, it is shown that the CNN trained for each object category does not generalize to other categories. Besides, the volumetric representation restricts the resolution of modeled shapes. The work by Lun et al. [2017] inputs category-specific planar sketches from canonical viewpoints (front, side, top) to a CNN with an encoder and thirteen decoders, each of which outputs the depth and normal maps for one of thirteen predefined viewpoints, which are then fused together into a 3D mesh.

Different from [Delanoy et al. 2017] and [Lun et al. 2017] that solve the generation of complete 3D shapes of trained categories, our work focuses on modeling freeform surfaces that are represented as depth maps, while also providing a multi-view fusion approach to combine the surfaces into full 3D models. By modeling a surface at a time using general geometric rules and learned priors for shape from sketch, our approach is agnostic to shape categories. However, we note that to break up a complete 3D shape into multiple surface patches to be modeled sequentially is not always straightforward to conceive for users, which we regard as a due price to pay for the category-free advantage. To help the user modeling, our multi-view interactive process allows the user to sketch in arbitrary views for different parts of the shape incrementally, with surfaces modeled in other views assisting the sketching in new view (Sec. 4).

Procedural and parametric models provide another kind of prior knowledge, which effectively reduces the modeling task to a mapping from sketches to model parameters. Many works learn the mapping from data, for modeling urban architectures [Nishida et al. 2016], faces [Han et al. 2017], and others [Huang et al. 2016]. These methods are tailored for the given parametric models and do not generalize to other freeform shapes.

Recent works directly reconstruct from 2D images the 3D shapes and scenes represented in depth and normal maps [Eigen and Fergus 2015; Tatarchenko et al. 2016; Wang et al. 2015a], point cloud [Fan et al. 2017] or volumetric grids [Choy et al. 2016; Tatarchenko et al. 2017; Wu et al. 2016], utilizing data-driven and CNN models. In this paper we focus on reconstructing high quality 3D shapes from sketches which contain much sparser information than images, and provide the user convenient control for 3D modeling.

3 SINGLE VIEW MODELING

In a single view, we recover depth and normal data from a sparse planar sketch. There are two primary challenges for this process. First, the sparse strokes in a sketch have different meanings, each affecting proximate regions of the corresponding 3D shape differently; we need to parse the strokes consistently and interpolate their data over the entire planar region to infer the 3D surface. To solve this problem, we rely on a CNN model to parse the different input lines automatically with minimal user specification, thus saving much user effort. See Fig. 2 for an example where the ridges and valleys are distinguished automatically from input unlabeled strokes.

Second, the 2D sketches have inherent ambiguity of what 3D shapes they represent, which can fail whatever powerful machine learning model that tries brute force regression of the reconstruction. Previous approaches resolve the ambiguity and build feasible models usually by restricting to shapes of common classes; as a result, such a model works well for the particular category it is trained for but does not generalize to others [Delanoy et al. 2017; Lun et al. 2017]. We instead strive for more general freeform shape modeling and focus on using geometric principles with optional user input to combat ambiguities.

To summarize, at the core of our single view modeling is a two-stage CNN regression model (Fig. 2):

- Given the input sketches, a first stage subnetwork (DFNet) regresses the flow field, a dense signal that describes the surface curvature directions and guides its reconstruction (Sec. 3.2).
- A second stage subnetwork (GeomNet) takes the sketch and flow field guidance, and predicts depth/normal maps, and a *confidence map* that shows how much ambiguity there is for each point of the input sketch (Sec. 3.3).

In addition, the user can further modify the surface and resolve ambiguity, by providing curvature hints over strokes, or depth values on sparse sample points; our CNN model is trained to utilize these optional inputs. Next we discuss the single view modeling method in detail.

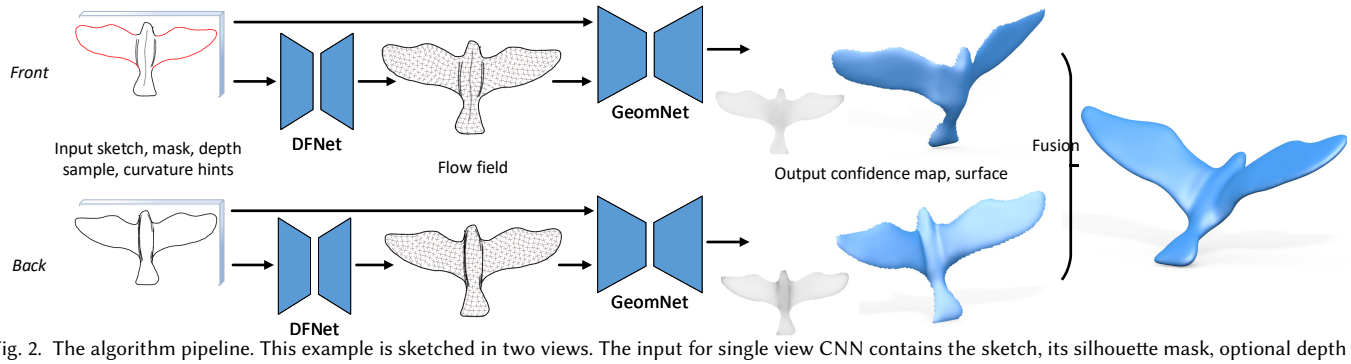
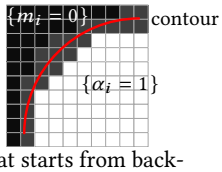


Fig. 2. The algorithm pipeline. This example is sketched in two views. The input for single view CNN contains the sketch, its silhouette mask, optional depth sample points and curvature hints (Sec. 3.1). An intermediate flow field is generated by the DFNet sub-network (Sec. 3.2), and further combined with input to feed into the GeomNet sub-network (Sec. 3.3) which predicts depth, normal and confidence maps. Part of the front surface contour (highlighted in red) is modified by re-sketching (Fig. 5) to lift the wings. Subsequent back view sketching reuses contour from the front surface. Finally, surfaces from the two views are fused into a complete shape (Sec. 4). See accompanying video for interaction process.

3.1 Input and output

Input. The input to single view modeling is primarily the 2D sketch represented by a gray scale image of a single channel, depicting the parallel projection of a 3D object on the current view. We require the distinction of contours and occluding contours from other strokes in the sketch, by forcing the (occluding) contours to be pure black and the other strokes to have nonzero gray scale values. This distinction is critical to clearly specifying where the depth and normal discontinues with (occluding) contours, while the rest strokes, including ridges/valleys, suggestive contours, curvature lines, etc., mainly describe the smooth shape variation. Indeed, such a line style complies with NPR literature [Rusinkiewicz and DeCarlo 2013], which shows that people use strong lines to emphasize boundaries, discontinuities and large curvatures. Our training data generation and modeling tool follows the style. Figs. 1,2,15 show examples of how the stroke style naturally leads to different surface variations. Note that for real user sketches, the input strokes represented by polylines are slightly smoothed to remove noisy artifacts before being rendered and fed to the network.



Binary masks are built out of the input sketches and used for network input and training loss computation. Sketch silhouette mask $\{m_i\}$ delineates the foreground object with $m_i = 1$ from background with $m_i = 0$ (inset, black), and is computed by a flooding process that starts from background pixels and stops at arbitrary sketch pixels; for sketches with holes, the user must pick a background point inside each hole to enable proper mask computation. $\{m_i\}$ is input to the network. Shape silhouette mask $\{m'_i\}$ for a training sample is the foreground pixel mask of rendered geometry; it usually differs from sketch silhouette mask $\{m_i\}$ by shrinking for a small margin, because the silhouette lines do have a width of several pixels. A signal filtering mask $\{\alpha_i\}$ (inset, white) removes invalid signals at pixels of discontinuous (occluding) contours, sharp features if any and background. $\{m'_i\}$ and $\{\alpha_i\}$ are used for training loss evaluation.

Depth samples are optional input to provide depth cues at certain points. They are encoded by a single channel map, with pixels of sample points having specified depth values and other pixels zero value (the 3D models have been positioned to ensure positive depth values). A sample point can be placed along contours and encoded by a single pixel, or inside the contour encoded by a 3x3 pixel patch.

Curvature hints are optional user inputs to specify the existence of sharp features where surface normal goes through abrupt change, and surface curvature, or how surface normal changes, across ridge, valley or sharp features. We use a map with three channels to encode the hints. One channel is a binary mask that is 1 for pixels of sharp features, and 0 otherwise. A second channel is a binary mask to indicate the existence of target curvatures whose values are given in the third channel. The target curvature value is $\mathbf{t}^{\perp} \cdot \Delta\mathbf{n} \in [-2, 2]$ for a pixel on stroke, where $\mathbf{t}$ is the unit tangent vector of the corresponding spatial curve on the training 3D model, $^{\perp}$ is to rotate the vector for 90° in the tangent plane, and $\Delta\mathbf{n}$ is the difference of surface normal vectors along $\mathbf{t}^{\perp}$ for a fixed step size (we use 3% of object bounding box diagonal length). Note the sign of curvature value denotes the convexity of surface across the stroke.

In total, the input map to our CNN for surface prediction has 6 channels.

Output. The output map consists of depth with 1 channel, normal vector with 3 channels, and confidence value with 1 channel. The output map is of the same resolution as input.

While the image size can be arbitrary because the CNN is fully convolutional, we have used 256×256 for a balance between the quality of 3D data and the training cost.

3.2 Flow field regression

Many lines in a sketch provide information about the surface bending (or curvature) directions, which is essential for recovering the 3D shape, as is shown in previous works [Bui et al. 2015; Iarussi et al. 2015; Li et al. 2017; Shao et al. 2012]. Rather than parsing the strokes and solving the flow field through heuristic algorithms and nonlinear optimizations as is done previously, we automate the entire process through CNN prediction.

Remark. It is worth mentioning that in [Groueix et al. 2018] an atlas parameterization is used to regulate the surface predicted by a decoder network. Similarly in our framework, the flow field provides for the second stage geometry reconstruction a locally smooth guidance at each point of the 2D domain, to enable the regulated propagation of data. It leads to more regular shapes with enhanced variations, as shown in Sec. 5.4.

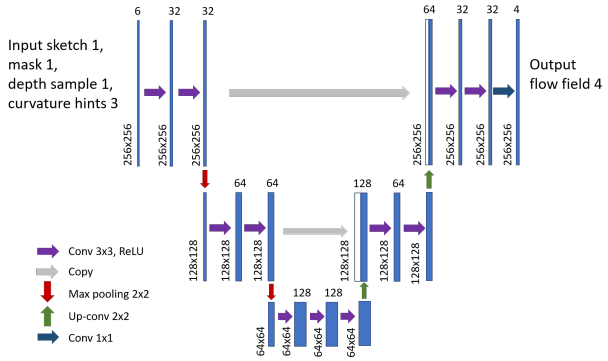


Fig. 3. Structure of DFNet. It is an encoder-decoder network with three domain resolutions. The input is a 6-channel image containing the sketch, its silhouette mask, and optional depth samples and curvature hints. The output is the 4-channel flow field. Numbers aside each feature map show its spatial size and the number of channels.

The DFNet sub-network (Fig. 3) is an encoder-decoder structure based on the commonly used U-Net [Ronneberger et al. 2015], with three levels of image resolutions. Given all inputs as discussed before, the network outputs a dense map with four channels $\mathbf{c}_i = (c_{i0}, c_{i1}, c_{i2}, c_{i3})$ for the i-th pixel, which encode a non-orthogonal 4-direction field $\mathbf{u}, \mathbf{v} \in \mathbb{C}$ through $\mathbf{u}^2 + \mathbf{v}^2 = c_0 + ic_1, \mathbf{u}^2\mathbf{v}^2 = c_2 + ic_3$, following [Diamanti et al. 2014]. Such a representation facilitates the comparison of directions at neighboring pixels by avoiding the need to match individual directions.

To train the network, we minimize the following loss function:

$$E_{field} = E_{data} + \gamma E_{var},$$

where

$$E_{data} = \frac{1}{\sum \alpha_i} \sum_i \alpha_i \left|\mathbf{c}_i - \mathbf{c}_i^0\right|$$

measures the absolute difference of regressed field from the projected curvature direction field $\{\mathbf{c}_i^0\}$ of the 3D shape, and

$$E_{var} = \frac{1}{\sum \alpha_i} \sum_i \alpha_i \sum_{k \sim i} |\mathbf{c}_i - \mathbf{c}_k|$$

is the total variation of the regressed field, with the k-th pixel the right or upper neighbor of the i-th pixel. The penalty of field total variation is necessary, because the sparse sketch lines representing surface bending directions do not fully capture the variance of the ground truth projected direction field, and we encourage a regular direction field to be derived from the lines instead. We have used $\gamma = 0.1$.

The signal filtering mask $\{\alpha_i\}$ resets loss at pixels covered by (occluding) contour lines, sharp features if any, and background. Reasons for using the mask are, first, the normal vectors for points on (occluding) contours face away to the sides and the projection of 3D curvature directions degenerates there, rendering the training data on these pixels unreliable, and second, the surface discontinues on (occluding) contour pixels and there is no expectation of direction field smoothness across them. When there are sharp features, their pixels have unreliable normal vectors and are excluded from loss computation as well. The output field is also masked by $\{\alpha_i\}$ when fed into the GeomNet sub-network.

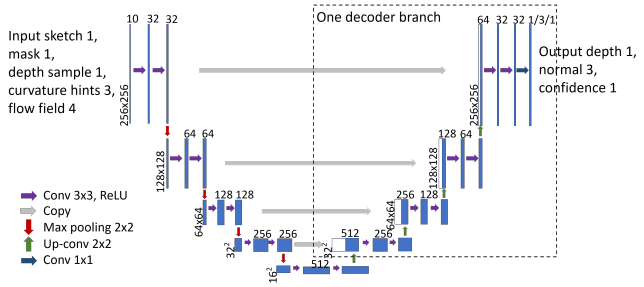


Fig. 4. Structure of GeomNet. It is an encoder-decoder network with five domain resolutions. The input is a 10-channel image containing the sketch, silhouette mask, optional depth samples and curvature hints, and the flow field. There are three decoder branches that share the same encoder, predicting depth, normal, and confidence respectively.

3.3 Robust flow-guided surface regression

The input sketch, silhouette mask, and optional depth samples and curvature hints are stacked together with the regressed flow field, and fed into the GeomNet sub-network for predicting normal and depth maps representing the 3D surfaces. GeomNet (Fig. 4) is an encoder-decoder structure with five levels of image resolutions and three decoder branches sharing the same encoder, which output normal, depth and the confidence maps respectively. To train the network, we minimize the following loss function:

$$E_{total} = E_{robust\_dn} + \beta E_{sample} + \lambda E_{reg}.$$

Robust data fitting. The first term measures how much the regressed depth d_i and normal vector $\mathbf{n}_i$ differ from the ground truth values $d_i^0, \mathbf{n}_i^0$ taken from the 3D shape:

$$E_{robust\_dn} = \frac{1}{\sum m_i'} \sum_i m_i' w_i^2 \left(\sigma_d (d_i - d_i^0)^2 + \sigma_n \|\mathbf{n}_i - \mathbf{n}_i^0\|^2\right) + m_i'(1 - w_i)^2,$$

where $\{w_i | 0 < w_i < 1\}$ is the confidence map.

A decoder branch outputs the confidence map through a final layer sigmoid activation. The function of confidence map is essentially a robust estimator [Black and Rangarajan 1996]: it tends to be 1 as required by the second term in E_{robust_dn} to enforce the learning of ground truth data, but will be relaxed if the regressed depth and normal cannot match ground truth. Intuitively, the confidence map estimates how much ambiguity there is for each point of the input sketch, because ambiguity is the ultimate cause of the mismatch for a well-trained geometry regression model. See Fig. 9 for example confidence maps showing the amount of ambiguity that agrees with expectations.

Although ambiguity is apparent to human observers, it is hard to quantify. Here the training of confidence map is *unsupervised*, avoiding the need for ground truth data generation. The benefit of confidence map is primarily to make learning more approachable by relaxing the depth and normal fitting loss over highly ambiguous regions. This is shown in the ablation test in Sec. 5.4. Moreover, during the fusion process where surfaces from multiple views are adapted into a complete 3D shape, it is useful for weighing the points so that we can preserve the reliable and change the rest (Sec. 4). Finally, it can be regarded as feedback to the user about where the

network prediction is uncertain and can use more input data to reduce ambiguity.

σ_d, σ_n normalize the errors of depth and normal regression. To estimate them, we train a straightforward regression of depth and normal without confidence map (Sec. 5.4), and collect the mean errors of the predicted depth and normal on the test dataset, based on which we get $\sigma_d = 920 \approx 1/(2{\times}0.0232^2)$, $\sigma_n = 430 \approx 1/(2{\times}0.0339^2)$.

Remark. The robust data fitting shares similarities with the framework of Bayesian deep learning [Kendall and Gal 2017], where uncertainties due to both the CNN model and data noise are modeled by variables very similar to the specific confidence map we used. However, since in our task the training data is generated from ground truth 3D shapes, the confidence map mostly models the uncertainty of CNN regression model, which intuitively originates from the ambiguity of mapping sketch to 3D shapes.

Depth sample constraint. If there are depth samples provided, the regressed depth map at sample points should match the specified depth values d_s^0, which we encourage through:

$$E_{sample} = \frac{1}{S}\sum_{s=1}^{S}(d_s - d_s^0)^2,$$

where S is the number of pixels covered by the sample points.

Depth and normal consistency. Finally, the regressed normal vectors and depth values should be consistent geometrically for a valid 3D shape, which is measured by:

$$E_{reg} = \frac{1}{\sum \alpha_i}\sum_i \alpha_i \left((\mathbf{n}_i \cdot \mathbf{t}_{ix})^2 + (\mathbf{n}_i \cdot \mathbf{t}_{iy})^2\right),$$

where $\mathbf{t}_{ix} = \left(1, 0, (d_j - d_i)/k\right), \mathbf{t}_{iy} = (0, 1, (d_l - d_i)/k)$ are tangent vectors at pixel i computed by finite difference with its right and upper neighbor pixels j, l, and $k = 0.00784$ is the pixel width mapped to the canonical scale of training data. $\{\alpha_i\}$ is used to reset loss at discontinuities.

We have used $\beta = 5$ to emphasize the matching to given sample points. Because initially during training the regressed normal and depth are far from being correct, we relax the consistency requirement, and increase it gradually; specifically, λ goes from $5{\times}10^{-4}$ to 2, multiplying 3.985 every 5k iterations during training which has a total of around 42k iterations. In real use cases, there is no shape mask $\{m_i'\}$ from ground truth data; the output maps are filtered with silhouette mask $\{m_i\}$ instead.

3.4 Data generation and network training

Data generation. The training samples are pairs of input, i.e. a sketch, its mask, optional depth samples and curvature hints, and the corresponding output, i.e. projected curvature direction field and depth/normal maps from the 3D shapes. Details of data generation are in the supplemental material; below we give an overview.

The freeform smooth shapes used for data generation are common models in Computer Graphics. They cover a wide range of categories, including animals, statues and characters, with details that can be reasonably depicted by uncluttered 2D sketches. To enable user specification of sharp features, we also generate models with sharp creases by iteratively applying rolling guided normal filtering [Wang et al. 2015b] to smooth shapes. By using this diverse set of objects, we make our geometric inference from sketches independent of specific shape categories and thus targeting generic freeform shapes.

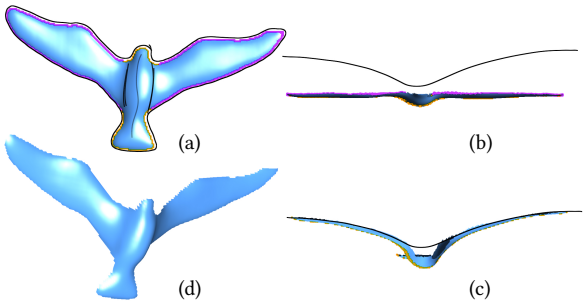


Fig. 5. Re-sketching contour in a side view to modify the shape. (a) the current surface overlaid with the sketch. Parts of the contour line to re-sketch are marked purple. (b) the surface seen in a side view, and a new profile curve to which the marked contour would match up. (c) (d) the surface updated with new depth cues.

The training samples are generated by *mimicking how humans sketch to depict 3D shapes*. Sketches are generated by NPR rendering with the program [Rusinkiewicz and DeCarlo 2013]. We select the parameters of NPR so that for a human observer it is easy to infer the 3D surface relatively well from the rendered 2D sketch. Viewpoints and zooming factors for rendering are evenly sampled in the valid range but filtered so that the observed surface patches are large enough and mostly front-facing, which comply with how users would generally pose the object. Optional depth samples are placed along portions of the silhouette curves to anchor the surfaces; during modeling, for many shapes the contour can simply be initialized to the drawing plane and tuned later by re-sketching (Sec. 4). Few (less than 5) additional depth samples are generated inside the contour, and placed over extreme depth values of the surface. Optional curvature hints are randomly generated for detected sharp features and ridge/valley curves of the 3D surface.

In total, there are 260k and 58k samples for training and test, respectively.

Network training. The network is trained in a two-stage process. First, we train the DFNet sub-network for 10 epochs, minimizing the loss function E_{field}. Next, we train the GeomNet sub-network with the loss function E_{total} for 10 epochs, while fixing the DFNet. Each network is implemented in Tensorflow, and optimized by Adam solver [Kingma and Ba 2014] with a fixed learning rate 10^{-3}, on a machine with 4 Nvidia GeForce 1080Ti GPUs. DFNet training takes 8 hours, and GeomNet 16 hours.

4 MULTIPLE VIEW MODELING

To model complete 3D shapes, multiple surfaces from different views have to be used. Sketching in multiple views can model different parts to be fused into the final complete shape. During the multi-view process, existing surfaces can assist the modeling of new parts by providing partial sketches and depth data at overlapping regions. In this section we discuss how to combine our single view modeling network with multi-view sketching, and the convenient user interactions enabled.

Re-sketching contour in a rotated view. The depth of contour curve from one viewpoint becomes a profile curve when seen through a

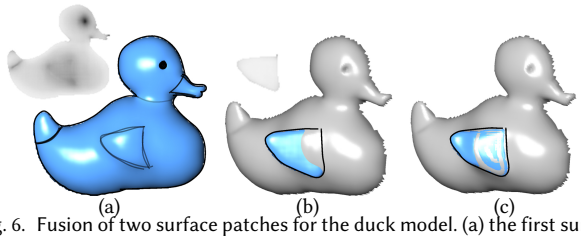


Fig. 6. Fusion of two surface patches for the duck model. (a) the first surface is sketched, with its confidence map shown at upper left corner. The duck wing is not easy to model in this view due to self-occlusion, with the confidence map showing lower value there. In a slightly rotated view (b), the duck wing is sketched properly, with high confidence shown at upper left corner. (c) the two surfaces are fused tightly at overlapping regions, biased toward the wing patch with higher confidence value.

rotated view, which makes it very convenient to modify the contour depth of the former view by re-sketching the profile curve in the latter view. An example is shown in Fig. 5: in the original view, part of the contour curve which we want to modify is marked (a); it is given a target re-sketched profile curve in another view (b), which is used as additional depth sample cues of CNN input for updating the surface (c)(d).

Sketching in a new view with assistance. When sketching in a new view, the surfaces from previous views project to the current view, which can assist the sketching and modeling in the current view. On one hand, salient NPR curves are generated for the existing surfaces, which the user can simply reuse. On the other hand, visible existing surfaces provide depth cues for the current surface prediction. For example, in Fig. 2, the contour curve along with its depth data of the front view surface is directly copied to the back view, where the user only needs to draw the inner strokes to model the back surface.

Multi-view fusion. In the final step, all surfaces are placed into appropriate positions and fused into a complete 3D shape. Proximate surfaces which have overlapping regions may not be consistent, which we resolve by the following quadratic optimization:

$$\min_{\{\mathbf{x}_i\}} \frac{1}{N}\sum_{i=1}^{N} -\omega_0 log(1-w_i)\cdot\left(\mathbf{x}_i-\mathbf{x}_i^0\right)^2+\omega_1\left(\Delta\mathbf{x}_i-\Delta\mathbf{x}_i^0\right)^2 + \frac{\omega_2}{|P|}\sum_{(i,j)\in P}\left(\mathbf{n}_i\cdot(\mathbf{x}_i-T_{ji}\mathbf{x}_j)\right)^2,$$

where N is the number of points from all views, $\mathbf{x}_i$ the 3D position of the i-th point in the coordinate frame of its local view, w_i the predicted confidence value, $\mathbf{x}_i^0$ the predicted position, $\mathbf{n}_i$ the predicted normal vector, $\Delta\cdot$ the Tutte Laplacian operator [Gotsman et al. 2003] for three coordinates built on the 3×3 neighborhood of a pixel (we use the simple Tutte weight because the domain is a regular pixel grid), and P the set of matched point pairs where the j-th point from its corresponding view is reprojected through frame transformation T_{ji} into the view of i-th point to have the same x, y coordinates as i.

To avoid erroneous matching that comes with viewing occlusions and imperfect partial reconstructions, we filter out a pair of matched points, if they have a distance above 10% of the smaller contour bounding box diagonal length of the two views containing i, j, or if their predicted normal vectors form an angle larger than 70°. The confidence value is mapped to log-scale to enforce the more reliable surfaces be preserved while less confident parts be

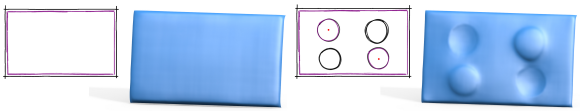


Fig. 7. A simple abstract shape sketched in process. To create this bumpy plane, the user can simply draw a plane first using contours and sharp features, and then add bumps and holes by drawing circles inside and using depth samples (red points) to push the holes under plane.

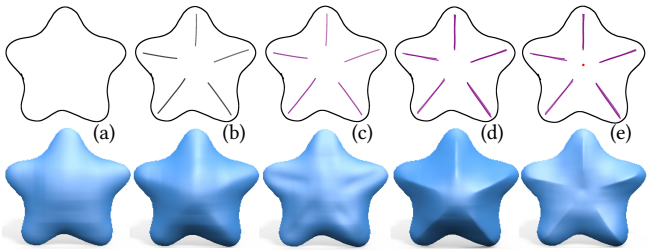


Fig. 8. A sequence of user sketching and editing and the predicted surfaces. (a) the contour only and its surface. (b) five inner strokes are drawn. (c) the inner strokes are given curvature hints that have negative signs. (d) the inner strokes are replaced with sharp features. (e) an inner depth sample (red point) is placed at the center and pulls the region flat.

adapted. The penalty of Laplacian deviation is to keep the predicted shapes as much as possible. We use $\omega_0 = 0.1, \omega_1 = 100, \omega_2 = 0.1$ to achieve a smooth adaptation process that preserves the critical surface variations.

The optimization problem is solved for 4 iterations; in each iteration, the point correspondences are updated and the linear least square problem is solved with an iterative Conjugate Gradient solver for 100 iterations. Fig. 6 shows an example of tightly fused patches. Note this fusion process can be largely accelerated if we use as variables a proper subset of the dense pixels which contain a lot of redundancy. This can be achieved through down-sampling, which we leave for future work.

Finally, a surface mesh is extracted from the fused point set by Screened Poisson Reconstruction [Kazhdan and Hoppe 2013], to which we pass along the adapted points and their normal vectors updated based on new positions.

5 RESULTS AND DISCUSSION

5.1 Results

To model a target freeform shape that is category-free and could be abstract, the interactive sketching process is generally quite intuitive: the user incrementally explores and draws to add more details and even surface patches in new views. Fig. 7 shows how to sketch a simple abstract bumpy plane by adding bumps and holes to a base plane. Fig. 8 shows the modeling of various stars through a sequence of elementary user inputs and edits, where the predicted surfaces respond to the strokes, curvature hints, sharp features and depth sample accordingly. More results can be found in Figs. 1, 2, 13, 15. The accompanying video shows real time modeling using our tool.

Surface patches are constructed by triangulating the pixel grids, and are lifted to predicted depth values and rendered with predicted normal vectors. For single surface modeling, the contour curve is by default provided with depth samples in the drawing plane unless otherwise specified. Re-sketched contour curves are marked in red; inner depth samples if any are also shown as red points.

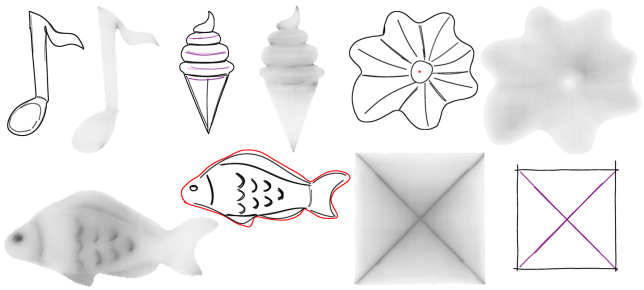


Fig. 9. Sample confidence maps and their sketches. The simple sketch for music note, the strokes with curvature hints (purple) for ice cream, and the depth sample (red point) for the curvy plate all lead to less ambiguity and brighter confidence maps. In contrast, over-sketched and cluttered local details as in fish and sharp features (purple) with unknown normal jump for the pyramid have more ambiguity and darker confidence maps. The result 3D shapes are shown in Figs. 1, 11, 15.

Strokes specified as sharp features or with target curvature values are marked in purple.

To better understand the behavior of robust data fitting loss, we visualize the confidence maps for representative examples with more or less ambiguous sketches (Fig. 9). As we would expect, simple sketches or the use of depth cues and curvature hints lead to low ambiguity and boost predicted confidence values. On the other hand, over-sketched and complex strokes that are relatively cluttered, and sharp features across which there is undetermined surface normal jump, can be more ambiguous and lower the confidence values.

Runtime. Single view surface prediction by a CNN forward pass is instantaneous (around 42ms), which allows for interactive feedback and smooth sketching exploration. Multiple view fusion when used is relatively slow due to the large number of variables for all valid pixels, but takes less than 10 seconds even for as many as five surface patches. Poisson reconstruction generally takes less than 3 seconds to extract a surface mesh from a point set. The timing is reported on a desktop PC with Intel Xeon 2.0GHz CPU and Nvidia GeForce GTX 980 GPU.

5.2 Comparison

Non-learning based methods. For many classical methods like [Igarashi et al. 1999; Nealen et al. 2007], the modeling process is an interactive and incremental session, where each single stroke defines a new boundary or feature curve in 3D that modifies the current base shape, by solving a predefined geometric smoothness prior like biharmonic equations. In comparison, for each view, our approach takes a 2D sketch as input and generates in one pass a detailed 3D shape that follows learned geometric priors. On one hand, such a difference in modeling paradigm makes our approach a complement to the previous methods as it can be used for efficient surface patch modeling by sketching in a single view. On the other hand, the learned geometric priors can be more reasonable than hand-crafted rules. In Fig. 10, we see that to model a bottle shape, with our method the 2D sketch is directly converted into a 3D surface (d), while for biharmonic surfaces, with boundary position and normal constraints, the result shape loses key features we want (b), and is overly rounded even when additional inner curve constraints are provided (c).

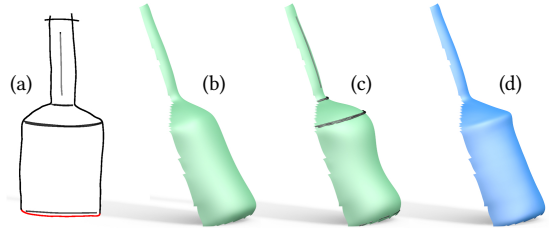


Fig. 10. Comparison with biharmonic surfaces. (a) the input sketch of a bottle shape. Red part of the contour is re-sketched to a half circle. (d) our predicted result. (b) using the boundary position and normal constraints taken from (d), the computed biharmonic surface cannot preserve key features of the bottle shape. (c) even if we further constrain the biharmonic surface with lifted spatial strokes (black curves) taken from (d), it is overly rounded and loses the desired characteristics.

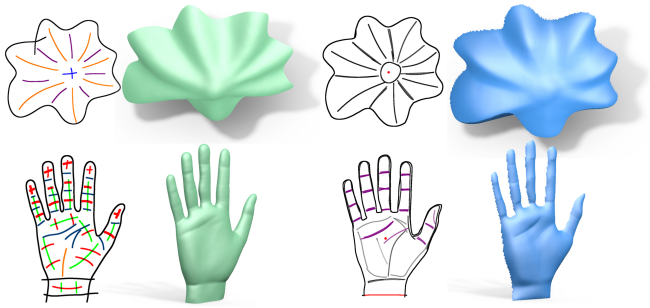


Fig. 11. Comparison with BendSketch [Li et al. 2017]. We recreate shapes (right) similar to those from their paper (left). While their method requires detailed annotations about the types of many more strokes (ridge, valley, curvature line, sharp feature, etc., shown through color) and tuning stroke curvature magnitudes (shown through line width), our method uses much more succinct and natural gray scale sketch, with few additional specifications like depth sample (red point) or sharp features (marked in purple). For our hand model, the red contour is re-sketched to be slightly off plane.

BendSketch [Li et al. 2017] reconstructs freeform surfaces with complex curvature variation patterns from a 2D sketch, but the type and meaning of each stroke must be specified by user as input, so that the predefined geometric rules can be applied properly. In comparison, thanks to our data-driven approach, our method parses a user's freehand sketching with more succinct strokes and very few annotations. Examples are shown in Fig. 11, where similar surface patches are created by both methods with very different sketches.

Learning based method. As reviewed in Sec. 2, most existing learning based methods model category-specific shapes, while our method targets generic freeform shapes. Still, here we compare with the multi-view decoder approach [Lun et al. 2017] which outputs intermediate depth and normal maps of predefined views to be fused into 3D shapes, on modeling one of their shape categories, characters, to demonstrate the impact of different network structures on result quality.

To make meaningful comparison, on the characters dataset published by [Lun et al. 2017], we use sketches in the front view as the only input. For their multi-view decoder network, the input is front view sketch image, and the output are thirteen depth and normal maps for views of the same number, which consist of one frontal view and twelve regularly sampled spherical views around the object. Their network is trained to minimize a data fitting loss

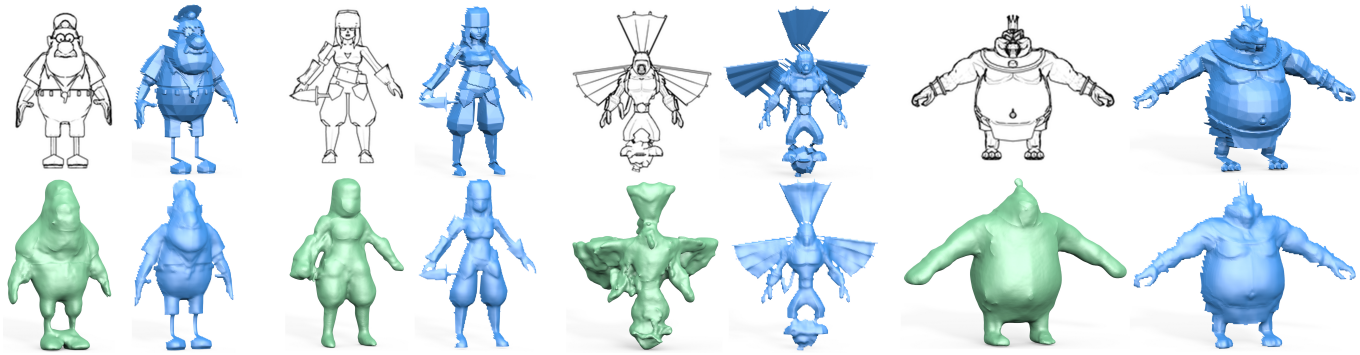


Fig. 12. Comparison with the multi-view decoder network [Lun et al. 2017], both networks trained and tested on the characters dataset. The input sketch for each test case is shown on the upper left, with ground truth front view depth map shown on the upper right. The multi-view fused complete 3D shapes by [Lun et al. 2017] are shown on the lower left in green, while our predicted front view surface refined by post-regularization are shown in blue on the lower right. Although our results are not complete 3D models, they capture more critical details intended by the sketches.

and an adversarial loss that captures category-specific properties. We remove the mask regression objective from their network for uniformity. For our network, the input front view sketch first goes through the stage-one flow field regression network, and then the sketch and flow field map together go through the stage-two geometry network to output depth and normal maps for the front view. Noticing that many of the 3D models in the dataset are low-poly shapes which lack smoothness, we relax the regularity weight to $\lambda = 0.01$ (Sec. 3.3) for training our network; since there is no stroke type data in the training set, we use a conservative signal filtering mask $\{\alpha_i\}$ that is zero for all stroke pixels and background. Ground truth curvature direction fields are generated using the depth maps in training data. We replace our basic UNet component with the smaller encoder-decoder structure used in their network, which suits the small dataset with 10k samples; as a result, our network is more than two times smaller than theirs. All networks are trained for 10 epochs to convergence.

Since in [Lun et al. 2017] the predicted depth and normal maps for multiple views are fused with a post-processing procedure that consists of regularization, Poisson reconstruction, and detail enhancement, to make a fair comparison, following [Nehab et al. 2005], we apply to our results a simple regularization that refines depth maps for more consistency with normal maps, which in training is relaxed as discussed above. The two networks are evaluated on the test dataset to produce full 3D shapes and front-view surfaces respectively. We compute the data fitting errors from the front view ground truth depth maps: to compare two depth maps, we first align them by translation so that they have common mean value, and then compute their absolute difference averaged by pixel numbers; for the fused 3D shapes which may have a different front-view silhouette than ground truth, we simply omit non-matching pixels from error counting. The angular errors between normal vectors of reconstructed surfaces and ground truth are also computed.

The depth/normal errors for their results are 0.0294/22.4°, and 0.0332/18.6° for ours. Visual results (Fig. 12) show our network predicts shapes that better capture important details conveyed by the input sketch. It demonstrates that our two-stage network with ambiguity estimation is beneficial even for category-specific reconstruction from sketches.

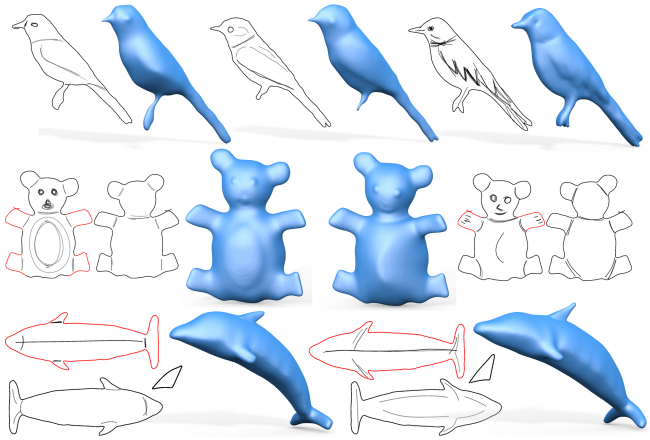


Fig. 13. Sketches of various styles drawn by the users to model target shapes in the evaluation test. Red contours are re-sketched to lift the teddy limbs or to bend the dolphin body. Bird shapes are made by one front sketch and completed by a symmetric back surface. Teddies are made by surfaces for front and back sketches. Dolphins are sketched in three views; the triangular sketches model the top fins. On average, the novice users without background in drawing or modeling made the bird, teddy and dolphin shapes in less than 5, 15 and 20 minutes respectively.

5.3 User evaluation

We invited five novice users with little knowledge or training about sketching and 3D modeling to evaluate our tool. We first introduce to the participants the background of how line sketches depict 3D shapes, and then show them how our tool works by sketching simple examples, to make them familiar with user interface. The training takes 20 minutes for each person. Then in the evaluation test, the users are asked to do three modeling tasks where they create three target shapes of different complexity. The target shapes, i.e. bird, teddy and dolphin, are given by reference images or 3D models rendered in another program.

Example sketches and 3D shapes created by the participants are shown in Fig. 13; more user creations and evaluations are in the supplemental material. The users sketch in different styles which appear natural to them. They managed to create interesting shapes in 5 minutes, 15 minutes, and 20 minutes for bird, teddy, and dolphin respectively. The bird models are all drawn in a single view, and made complete by a symmetric reflection of the front surface. The

teddy models are drawn in front and back views, where the front surface contour is copied to the back for assistance. The dolphin models are drawn in top, bottom and side views; the side view only contains a triangular sketch modeling the top fin which is fused with the other two patches to complete the shape.

Feedback from the participants is mostly positive. After 20 mins training, users find the sketches easy to perceive for depicting the shapes. After they have learned to use the tool, they like to take their time and explore variations. Moreover, the users find sketching in multiple views to make complete shapes easy to use.

5.4 Ablation study

To further test the impact of our network structure, we have done three additional experiments for alternative structures:

(1) A straightforward regression of depth and normal data from input sketches and depth samples, with loss function $E_{dn} = \frac{1}{\sum m_i'} \sum_i m_i' \left(0.76 \times (d_i - d_i^0)^2 + \|\mathbf{n}_i - \mathbf{n}_i^0\|^2\right)$. The coefficient 0.76 for depth fitting term is derived from the range of depth difference $[0, 2.6]$ and the norm range of normal vector difference $[0, 2]$.
(2) A modified regression with losses measuring shape regularity and depth sample constraints, i.e. $E_{dn} + \beta E_{sample} + \lambda E_{reg}$.
(3) A further modified regression with confidence map and the loss function E_{total}, but without the first stage network or intermediate flow field.

These networks have the same structure as GeomNet, i.e. a five-level encoder-decoder with separate decoders for different outputs. They are trained on the same dataset as presented before for 10 epochs to convergence (Sec. 3.4).

The trained networks are evaluated on the test dataset. Representative visual results are shown in Fig. 14. Average test errors are reported in Table. 1, where again the depth error is computed after we align the predicted and ground truth depth maps so that they have common mean value. The normal errors are angles in degrees measured between normal vectors derived from the depth maps and ground truth. In addition to the above configurations of networks, we apply Laplacian smoothing [Desbrun et al. 1999] ($\lambda = 1$, 5 iterations) to the predicted results of (1), to check if this simple post-processing can improve result quality.

The results for all configurations match what we expected. The naive fitting network (1) produces results with small errors in depth but large errors in normal, which visually translates into the globally irregular and locally noisy surfaces (Fig. 14). Laplacian smoothing may reduce noise but does not improve the overall shape. With the additional objective of depth/normal consistency, (2) has much reduced normal error but increased depth error than (1); visually the result shapes are locally smoothed but overall distorted. Regression with confidence map modeling ambiguity (3) not only greatly improves surface regularity but also has lower errors in both depth and normal than (2). Still the full network with flow field guidance has depth error almost equal to naive predictions, but much smaller normal error. Visually, as shown in Fig. 14, the result surfaces by full network are more regular with rich variations faithfully capturing sketch intentions. In addition, while all networks produce better results when given more depth cues, the full network is most resilient to the lack of depth cues, which we argue is due to the dense flow field regulating the surface prediction.

Network	(1)	(1) smoothed	(2)	(3)	full
Depth ($\times 10^{-2}$)	2.14	2.19	2.36	2.24	2.18
Normal (degrees)	11.59	11.70	9.86	9.24	9.07

Table 1. Average test errors for comparing networks. (1) is a naive regression of depth and normal. (1) smoothed applies Laplacian smoothing to the results of (1). (2) augments (1) with regular and depth constraints in loss function. (3) uses the same loss function as ours with confidence map. Full is the complete two-stage network.

6 CONCLUSION

In this paper we presented a sketch-based freeform surface modeling method that uses a CNN model to infer from sparse 2D sketches depth and normal maps representing the surfaces. Compared with traditional methods, the new learning based approach enables direct inference from 2D sketches with more succinct lines and much fewer user annotations. Different from existing learning-based methods which resolve the 2D-to-3D ambiguity by modeling category-specific shapes only, our approach targets generic freeform shapes and handles the ambiguity by using basic geometric and statistical rules realized in our novel network structure.

Our network has two stages: in the first stage, the network regresses a dense flow field from sketch which regularizes subsequent geometry reconstruction; in the second stage, given the input sketch and flow field, the network predicts depth and normal maps, along with a confidence map that quantifies the amount of ambiguity on each point, which is trained unsupervised and makes the geometry regression more robust. Our network also allows user modification of a surface by providing depth sample points, sharp features and curvature hints. To train the network, we have generated a dataset through NPR rendering and closely mimicking how users would actually draw sketches to depict 3D shapes. Finally, multiple view sketching provides convenient tools like contour re-sketching, and enables the modeling of complete 3D shapes by fusing surfaces for different parts of an object.

We have made validations, comparisons, user evaluations and ablation tests, which show that the proposed approach is a new way for modeling diverse freeform shapes by drawing intuitive and expressive sparse 2D sketches.

Limitations and future work. Our approach models a surface patch at each time, which is a paradigm unsuitable for highly structured and symmetric shapes like CAD models, which can be more conveniently defined by sketching complete and regular wireframes.

We used a drawing canvas of fixed resolution 256×256. For complex shapes, it is natural to model different level-of-details in multiple resolutions. We would like to extend our method and user interface to implement such a multi-scale framework for modeling both the overall shape and fine details flexibly.

In this work, we have considered sketches that have sparse lines as input. However, dense strokes depicting shading cues are also widely used by artists to enhance the 3D perception. In the future, we want to handle these sketching styles for 3D modeling as well.

Tackling ambiguity is a pervasive issue in data-driven and machine learning tasks, where we believe the application of robust estimator can be generally helpful.

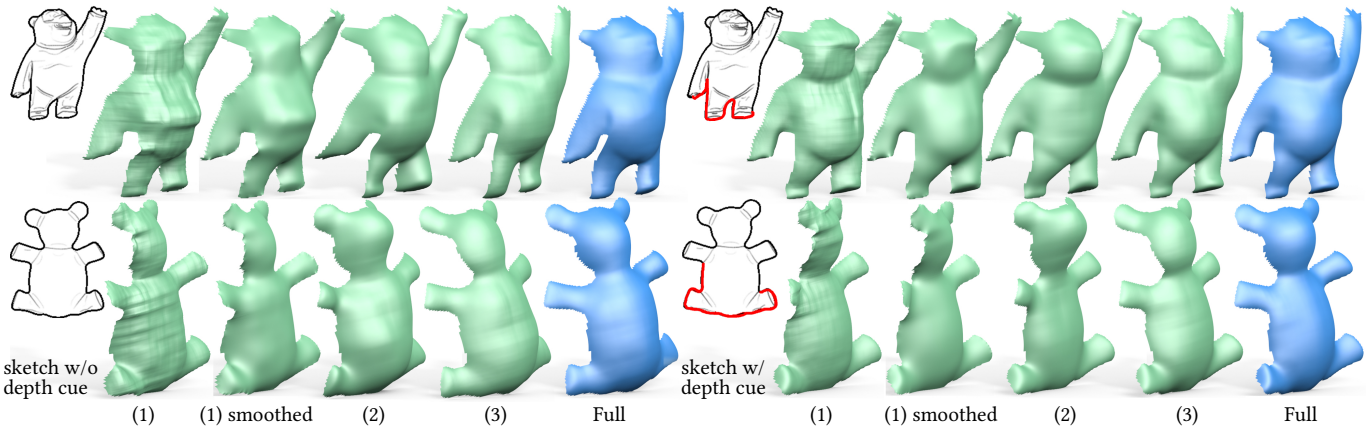


Fig. 14. Predicted surfaces by different networks for ablation test. Left columns show results for input sketches without any depth samples, right columns with depth samples (marked in red) along portions of the contour curves. The sketches are rendered from existing test models. (1) is the naive regression of depth and normal. (1) smoothed applies Laplacian smoothing to network predictions. (2) augments (1) with regularity and depth sample constraints in loss function. (3) further uses the robust data fitting with confidence map. Full network has the complete two-stage structure. (1) produces very noisy results, which are still distorted after Laplacian smoothing. (2) reduces noise, but the overall shapes are distorted. (3) predicts more regular surfaces. But the full network further improves regularity, and has richer surface variations capturing the meaning of input strokes faithfully (e.g. see head of top bear). All networks perform better when given depth cues, but the full network is most resilient to the lack of depth samples.

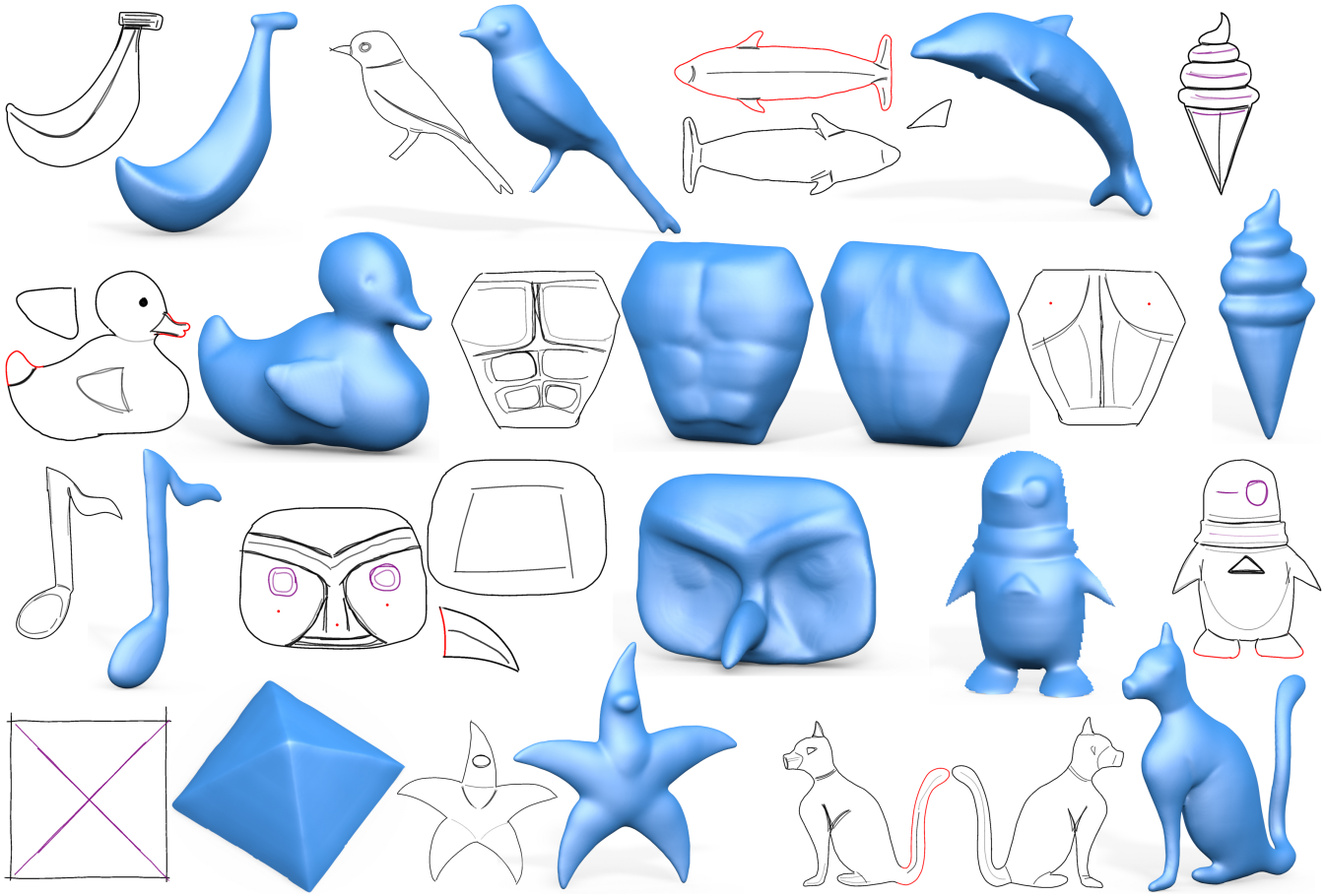


Fig. 15. More sketches and corresponding shapes. Red contours are re-sketched to provide additional boundary depth data. Inner depth samples (red points) are used for the torso back and owl face. Sharp features (purple) are used in pyramid, penguin, owl face. Curvature hints are used for ice cream. Dolphin fin, duck wing, and owl nose are modeled by small triangle patches in their own views and then fused with the other parts. Penguin and pyramid are single patches; the rest are complete shapes.

ACKNOWLEDGMENTS

We thank the creators of utilized 3D datasets, the user evaluation participants, the reviewers for their suggestions and Timothy Jeruzalski for video narration. The work of Wenping Wang was partially supported by the National Basic Research Program of China (2011CB302400), NSFC (61272019) and the Research Grant Council of Hong Kong (717012, 17209815, 17211017). The work of Alla Sheffer was supported by NSERC.

REFERENCES

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: As-natural-as-possible Sketching System for Creating 3D Curve Models. In *UIST*. 151–160.

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

Michael J. Black and Anand Rangarajan. 1996. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV* 19, 1 (1996).

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

Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 2013. 3-Sweep: Extracting Editable Objects from a Single Photo. *ACM Trans. Graph. (SIGGRAPH ASIA)* 32, 6 (2013), 195:1–195:10.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *ECCV*.

Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. 2008. Where Do People Draw Lines? *ACM Trans. Graph. (SIGGRAPH)* 27, 3 (Aug. 2008).

Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. 2009. How Well Do Line Drawings Depict Shape?. In *ACM Trans. Graph. (SIGGRAPH)*, Vol. 28.

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive Contours for Conveying Shape. *ACM Trans. Graph. (SIGGRAPH)* 22, 3 (2003).

Johanna Delanoy, Adrien Bousseau, Mathieu Aubry, Phillip Isola, and Alexei A. Efros. 2017. What You Sketch Is What You Get: 3D Sketching using Multi-View Deep Volumetric Prediction. *arXiv preprint arXiv:1707.08390* (2017).

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *SIGGRAPH*. 317–324.

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing N-PolyVector Fields with Complex Polynomials. *SGP* 33, 5 (2014), 1–11.

D. Eigen and R. Fergus. 2015. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. In *ICCV*. 2650–2658.

Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. 2012. Sketch-based Shape Retrieval. *ACM Trans. Graph.* 4 (July 2012).

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*.

Yotam Gingold, Takeo Igarashi, and Denis Zorin. 2009. Structured Annotations for 2D-to-3D Modeling. *ACM Trans. Graph. (SIGGRAPH ASIA)* 28, 5 (2009), 148:1–148:9.

Craig Gotsman, Xianfeng Gu, and Alla Sheffer. 2003. Fundamentals of Spherical Parameterization for 3D Meshes. *ACM Trans. Graph. (SIGGRAPH)* 22, 3 (2003).

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*.

Xuekun Guo, Juncong Lin, Kai Xu, Siddhartha Chaudhuri, and Xiaogang Jin. 2016. CustomCut: On-demand Extraction of Customized 3D Parts with 2D Sketches. *Computer Graphics Forum* 35, 5 (2016), 89–100.

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

Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *SIGGRAPH*.

Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. 2016. Shape Synthesis from Sketches via Procedural Models and Convolutional Networks. *IEEE. T. Vis. Comput. Gr.* 22, 10 (2016), 1.

Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Trans. Graph. (SIGGRAPH)* 34, 3 (2015), 24:1–24:16.

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

Pushkar Joshi and Nathan A. Carr. 2008. Repoussé: Automatic Inflation of 2D Artwork. In *SBIM (SBM'08)*.

Tilke Judd, Frédo Durand, and Edward H. Adelson. 2007. Apparent ridges for line drawing. *ACM Trans. Graph. (SIGGRAPH)* 26, 3 (2007), 19.

Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. 2015. Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. *ACM Trans. Graph.* 34, 5, Article 155 (2015), 12 pages.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3, Article 29 (July 2013), 13 pages.

Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In *NIPS*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

Jeehyung Lee and Thomas Funkhouser. 2008. Sketch-based Search and Composition of 3D Models. In *SBIM (SBM'08)*. 97–104.

Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2017. BendSketch: Modeling Freeform Surfaces Through 2D Sketching. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 125 (July 2017), 14 pages.

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

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Trans. Graph. (SIGGRAPH)* 26, 3 (2007).

Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. 2005. Efficiently Combining Positions and Normals for Precise 3D Geometry. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (July 2005), 536–543.

Gen Nishida, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive Sketching of Urban Procedural Models. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016), 130:1–130:11.

L. Olsen, F. Samavati, and J. Jorge. 2011. NaturaSketch: Modeling from Images and Natural Sketches. *IEEE Comput. Graph. Appl. Mag.* 31, 6 (2011), 24–34.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, 234–241.

Szymon Rusinkiewicz and Doug DeCarlo. 2013. Real-time suggestive contours. (2013). http://www.cs.princeton.edu/gfx/proj/sugcon/

Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic Drawing of 3D Scaffolds. *ACM Trans. Graph. (SIGGRAPH ASIA)* 28, 5 (2009), 149:1–149:10.

Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A. Jorge. 2005. ShapeShop: Sketch-Based Solid Modeling with BlobTrees. In *SBIM*.

Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: Shading Concept Sketches Using Cross-section Curves. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012), 45:1–45:11.

Alex Shtof, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. 2013. Geosemantic Snapping for Sketch-Based Modeling. *Comput. Graph. Forum (EG)* 32, 2 (2013), 245–253.

H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *ICCV*. 945–953.

Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. 2018. Interactive Sketch-Based Normal Map Generation with Deep Neural Networks. In *ACM i3D*.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2016. Multi-view 3D Models from Single Images with a Convolutional Network. In *ECCV*. 322–337.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2017. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *arXiv preprint arXiv:1703.09438* (2017).

Fang Wang, Le Kang, and Yi Li. 2015c. Sketch-based 3D shape retrieval using Convolutional Neural Networks. *CVPR* (2015), 1875–1883.

Peng-Shuai Wang, Xiao-Ming Fu, Yang Liu, Xin Tong, Shi-Lin Liu, and Baining Guo. 2015b. Rolling Guidance Normal Filter for Geometric Processing. *ACM Trans. Graph. (SIGGRAPH ASIA)* 34, 6, Article 173 (2015), 173:1–173:9 pages.

X. Wang, D. F. Fouhey, and A. Gupta. 2015a. Designing deep networks for surface normal estimation. In *CVPR*. 539–547.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*. 82–90.

Xiaohua Xie, Kai Xu, Niloy J. Mitra, Daniel Cohen-Or, Wenyong Gong, Qi Su, and Baoquan Chen. 2013. Sketch-to-Design: Context-Based Part Assembly. *Comput. Graph. Forum* 32, 8 (2013), 233–245.

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 Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 131:1–131:13.

Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: Sketch-based Co-retrieval and Co-placement of 3D Models. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013), 123:1–123:15.

C. K. Yeh, S. Y. Huang, P. K. Jayaraman, C. W. Fu, and T. Y. Lee. 2016. Interactive High-Relief Reconstruction for Organic and Double-sided Objects from a Photo. *IEEE. T. Vis. Comput. Gr.* 99 (2016), 1–1.

Li Zhang, G. Dugas-Phocion, J. S. Samson, and S. M. Seitzt. 2001. Single view modeling of free-form scenes. In *CVPR*, Vol. 1. I–990–I–997.