3D Sketching using Multi-View Deep Volumetric Prediction

Johanna Delanoy
Inria Université Côte d'Azur

Mathieu Aubry
LIGM (UMR 8049), Ecole des Ponts

Phillip Isola
OpenAI

Alexei A. Efros
UC Berkeley

Adrien Bousseau
Inria Université Côte d'Azur

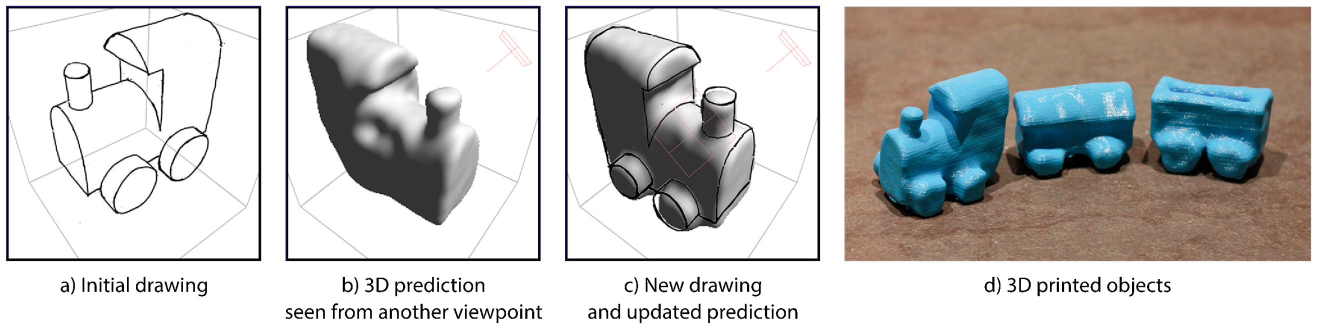

a) Initial drawing
b) 3D prediction seen from another viewpoint
c) New drawing and updated prediction
d) 3D printed objects

Figure 1: Our sketch-based modeling system can process as little as a single perspective drawing (a) to predict a volumetric object (b). Users can refine this prediction and complete it with novel parts by providing additional drawings from other viewpoints (c). This iterative sketching workflow allows quick 3D concept exploration and rapid prototyping (d).

ABSTRACT

Sketch-based modeling strives to bring the ease and immediacy of drawing to the 3D world. However, while drawings are easy for humans to create, they are very challenging for computers to interpret due to their sparsity and ambiguity. We propose a data-driven approach that tackles this challenge by learning to reconstruct 3D shapes from one or more drawings. At the core of our approach is a deep convolutional neural network (CNN) that predicts occupancy of a voxel grid from a line drawing. This CNN provides an initial 3D reconstruction as soon as the user completes a single drawing of the desired shape. We complement this single-view network with an *updater CNN* that refines an existing prediction given a new drawing of the shape created from a novel viewpoint. A key advantage of our approach is that we can apply the updater iteratively to fuse information from an arbitrary number of viewpoints, without requiring explicit stroke correspondences between the drawings. We train both CNNs by rendering synthetic contour drawings from hand-modeled shape collections as well as from procedurally-generated abstract shapes. Finally, we integrate our CNNs in an interactive modeling system that allows users to seamlessly draw an object, rotate it to see its 3D reconstruction, and refine it by re-drawing from another vantage point using the 3D reconstruction as guidance.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**;

KEYWORDS

sketch-based modeling, deep learning, 3D reconstruction, line drawing

ACM Reference Format:
Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A. Efros, and Adrien Bousseau. 2018. 3D Sketching using Multi-View Deep Volumetric Prediction. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The ambition of sketch-based modeling is to bring the ease and immediacy of sketches to the 3D world to provide *"an environment for rapidly conceptualizing and editing approximate 3D scenes"* [Zeleznik et al. 1996]. However, while humans are extremely good at perceiving 3D objects from line drawings, this task remains very challenging for computers. In addition to the ill-posed nature of 3D reconstruction from a 2D input, line drawings lack important shape cues like texture and shading, are often composed of approximate sketchy lines, and even when multiple drawings of a shape are available, their level of inaccuracy prevents the use of geometric algorithms like multi-view stereo. We introduce a data-driven sketch-based modeling system that addresses these challenges by *learning* to predict 3D volumes from one or several freehand bitmap drawings. Our approach builds on the emerging field of generative deep networks, which recently made impressive progress on image [Chen and Koltun 2017] and shape synthesis [Fan et al. 2017] but has been little used for interactive creative tasks.

Figure 1 illustrates a typical modeling session with our system. The user starts by drawing an object from a 3/4 view, which is the viewpoint preferred by designers to illustrate multiple sides of a shape in a single drawing. Thanks to training on a large collection of 3D shapes, our approach produces a complete volumetric reconstruction of the object, including occluded parts. This initial reconstruction allows the user to rotate the object and inspect it from a different vantage point. The user can then either re-draw the object from this new viewpoint to correct errors in the reconstruction, or move on to drawing new parts of the object. In both cases, the temporary 3D reconstruction acts as a reference that significantly helps the user create new drawings of the 3D shape. Since all interactions occur in a shared 3D space, this workflow provides us with multiple registered drawings of the object along with their respective calibrated cameras, which form the input to our 3D reconstruction algorithm.

At the core of our system are deep convolutional neural networks (CNNs) that we train to predict occupancy in a voxel grid, given one or several contour drawings as input. These CNNs form a flexible and robust 3D reconstruction engine that can interpret bitmap drawings without requiring complex, hand-crafted optimizations [Lipson and Shpitalni 1996; Xu et al. 2014] nor explicit correspondences between strokes in multiple views [Bae et al. 2008; Rivers et al. 2010]. However, applying deep learning to sketch-based modeling raises several major new challenges. First, we need a network architecture capable of fusing the information provided by multiple, possibly inconsistent, drawings. Our solution combines a single-view network, which generates an initial reconstruction from one drawing, with an *updater network* that iteratively refines the prediction as additional drawings are provided. This iterative strategy allows us to handle drawings created from an arbitrary number of views, achieving a continuum between single-view [Gingold et al. 2009] and multi-view [Bae et al. 2008] sketch-based modeling systems.

The second challenge we face is access to training data, as collecting thousands of hand-drawings registered with 3D objects would be very costly and time consuming. Similarly to prior data-driven approaches [Eitz et al. 2012b; Huang et al. 2016; Nishida et al. 2016; Xie et al. 2013], we alleviate the need for collecting real-world drawings by generating *synthetic* line drawings from 3D objects using non-photorealistic rendering. This allows us to easily adapt our system to the design of different types of objects by generating training data that is representative of such objects. We first illustrate this capability by training an instance of our system with a dataset of chairs, and another instance with a dataset of vases. We then target the design of more general objects by training our system with abstract shapes assembled from simple geometric primitives (cuboids, cylinders). We used this latter instance to model a variety of man-made objects and show that it generalizes well to unseen object categories. Finally, we describe how to co-design the training data and the user interface to reduce ambiguity in the prediction. In particular, we restrict viewpoints for the first drawing to avoid depth ambiguity for the single-view network, while we allow greater freedom for the subsequent drawings that are handled by the updater network.

Once trained, our system can generate a coherent multi-view prediction in less than a second, which makes it suited for interactive modeling. One restriction of our current implementation is that the resolution of the voxel grid hinders the recovery of thin structures. We thus target quick 3D design exploration rather than detailed modeling.

In summary, we introduce an interactive sketch-based modeling system capable of reconstructing a 3D shape from one or several freehand bitmap drawings. In addition to the overall system, we make the following technical contributions[1]:

- An iterative *updater network* that predicts coherent 3D volumes from multiple drawings created from different viewpoints .
- A multi-view drawing interface that we co-design with our synthetic data to help users create drawings similar to the ones used for training.

Note that our approach is modular and could adapt to other drawing techniques and shapes than the ones used in this paper.

2 RELATED WORK

Our work builds on recent progress in deep learning to tackle the long standing problem of sketch-based modeling. We refer the interested reader to recent surveys for extended discussions of these two fields [Cordier et al. 2016; Olsen et al. 2009; Srinivas et al. 2016].

2.1 Sketch-based modeling

The problem of creating 3D models from line drawings has been an active research topic in computer graphics for more than two decades [Igarashi et al. 1999; Lipson and Shpitalni 1996; Zeleznik et al. 1996]. While sketching is one of the most direct ways for people to represent imaginary 3D objects, recovering 3D information from 2D strokes poses significant challenges since an infinity of 3D shapes can potentially re-project on the same drawing [Barrow and Tenenbaum 1981]. Various approaches have been proposed to tackle the inherent ambiguity of this inverse problem.

Constrained-based approaches assume that the lines in a drawing represent specific shape features, from which geometric constraints can be derived and imposed in an optimization framework. Popular constraints include surface orientation along smooth silhouettes [Malik and Maydan 1989], orthogonality and parallelism of edges on polyhedral models [Lipson and Shpitalni 1996], symmetry [Cordier et al. 2013], and surface developability [Jung et al. 2015] among others. However, the assumptions made by these methods often restrict them to specific classes of shapes, or specific drawing techniques such as polyhedral scaffolds [Schmidt et al. 2009], curvature-aligned cross-sections [Shao et al. 2012; Xu et al. 2014] or cartoon isophotes [Xu et al. 2015]. In addition, most of these methods require clean vector drawings as input to facilitate the detection of suitable constraints, as well as to compute the various energy terms that drive the optimization. Unfortunately, converting rough sketches into clean vector drawings is a difficult problem in its own right [Favreau et al. 2016], while methods capable of directly recovering 3D information from noisy drawings are prohibitively

[1]Networks and databases are available online at https://ns.inria.fr/d3/3DSketching/

expensive [Iarussi et al. 2015]. In this work, we bypass all the challenges of defining, detecting and optimizing for multiple geometric constraints by training a deep convolutional neural network (CNN) to automatically predict 3D information from bitmap line drawings.

Interactive approaches reduce ambiguity in 3D reconstruction by leveraging user annotations. Single-image methods allow users to create 3D models from existing imagery by snapping parametric shapes to image contours [Chen et al. 2013; Shtof et al. 2013] or by indicating geometric constraints such as equal length and angle, alignment and symmetry [Gingold et al. 2009] or depth ordering [Sýkora et al. 2014]. Other methods adopt an incremental workflow where users progressively build complex shapes by drawing, modifying and combining simple, easy to reconstruct 3D parts. Existing systems differ in the type of assumptions they make to reconstruct intermediate shapes from user strokes, such as smooth shapes inflated from silhouettes [Igarashi et al. 1999; Nealen et al. 2007], symmetric or multi-view pairs of 3D curves related by epipolar constraints [Bae et al. 2008; Orbay and Kara 2012], curves lying on pre-defined planes or existing surfaces [Bae et al. 2008; Zheng et al. 2016], visual hulls carved from orthogonal viewpoints [Rivers et al. 2010]. The main drawback of such methods is that users have to mentally decompose the shape they wish to obtain, and construct it by following a carefully ordered series of sketching operations, often performed from multiple viewpoints. In contrast, while our system supports incremental modeling, our CNN-based reconstruction engine does not rely on restrictive assumptions on the drawn shapes and allows users to draw a complete object from one viewpoint before visualizing and refining it from other viewpoints.

Data-driven approaches exploit large collections of 3D objects to build priors on the shapes that users may draw. Early work focused on retrieving complete objects from a database [Eitz et al. 2012b; Funkhouser et al. 2003], which was later extended to part-based retrieval and assembly [Lee and Funkhouser 2008; Xie et al. 2013] and to parameter estimation of pre-defined procedural shapes [Huang et al. 2016; Nishida et al. 2016]. While our approach also learns shape features from object databases, we do not require these objects to be expressible by a known parametric model, nor be aligned and co-segmented into reconfigurable parts. Instead, our deep network learns to generate shapes directly from pairs of line drawings and voxel grids, which allows us to train our system using both existing 3D model databases and procedurally-generated shapes. Our approach is also related to the seminal work of Lipson and Shpitalni [2000], who used a database of random polyhedrons to learn geometric correlations between 2D lines in a drawing and their 3D counterpart. The considered correlations include the angles between pairs and triplets of lines, as well as length ratios. These priors are then used to evaluate the quality of a 3D reconstruction in a stochastic optimization. In a similar spirit, Cole et al. [2012] generate a large number of abstract blobs to serve as exemplars for a patch-based synthesis algorithm that converts line drawings into normal maps. While we build on these initial attempts, deep learning alleviates the need for custom feature extraction and optimization and allows us to handle a wider diversity of shapes. In addition, we integrate our 3D reconstruction engine in an interactive system capable of fusing information from multiple sketches drawn from different viewpoints.

2.2 Deep learning

Our work is motivated by the recent success of deep convolutional neural networks in solving difficult computer vision problems such as image classification [Krizhevsky et al. 2012], semantic segmentation [Long et al. 2015], depth and normal prediction [Eigen and Fergus 2015; Wang et al. 2015a]. In particular, our single-view volumetric reconstruction network follows a similar encoder-decoder architecture as depth prediction networks, although we also propose a multi-view extension that iteratively refines the prediction as new sketches are drawn by the user. This extension is inspired by iterative networks that implement a feedback loop to impose structural constraints on a prediction, for instance to refine hand [Oberweger et al. 2015] and human pose [Carreira et al. 2016].

Our architecture also shares similarities with deep networks tailored to multi-view 3D reconstruction. Choy et al. [2016] train a recurrent neural network (RNN) to predict a voxel reconstruction of an object from multiple uncalibrated photographs. Similarly, our iterative updater network can be seen as a recurrent network that is unrolled to simplify training. In addition, our modeling interface provides us with calibrated cameras by construction, since we know from which viewpoint each drawing is created. Unrolling the network allows us to apply the camera transformations explicitly as we iterate over each viewpoint. Ji et al. [2017] describe a multi-view reconstruction network that fuses two aligned voxel grids, each being filled with the color rays originating from the pixels of two calibrated input views. Their method extends to more than two views by averaging the predictions given by multiple pairs of views. Our updater network follows a similar strategy of implicitly encoding the camera orientation in the voxel grid. However, we iterate our updater over all drawings, one at a time, rather than combining multiple pairwise predictions at once. This design choice makes our method more sensitive to the order if which the drawings are created.

While CNNs have been mostly applied to photographs, they have also demonstrated impressive performances on tasks similar to ours, such as sketch cleanup [Simo-Serra et al. 2016], sketch colorization [Sangkloy et al. 2017], sketch-based retrieval [Sangkloy et al. 2016; Wang et al. 2015b], and sketch-based modeling of parametric shapes [Han et al. 2017; Huang et al. 2016; Nishida et al. 2016]. CNNs have also recently achieved promising results in *synthesizing* images [Chen and Koltun 2017; Nguyen et al. 2016; Park et al. 2017; Yan et al. 2016] and even 3D models [Dosovitskiy et al. 2016; Fan et al. 2017; Li et al. 2017; Wu et al. 2016, 2015] from low-dimensional feature vectors and attributes. We pursue this trend by training a deep network to generate voxelized objects from line drawings, offering precise user control on the shape being generated.

Two recent methods with similar goals have been developed concurrently to ours. First, Liu et al. [2017] combine a voxel sculpting interface with a generative network to project the coarse voxel shapes modeled by the user onto a manifold of realistic shapes. We see our sketch-based interface as an alternative to voxel-sculpting. Second, Lun et al. [2017] propose a method to reconstruct a 3D object from sketches drawn from one to three orthographic views. We share several ideas with this latter work, such as training with synthetic drawings and predicting 3D shapes from multiple views. On the one hand, Lun et al. achieve finer reconstructions than ours by extracting a 3D surface from multiple depth maps rather than from a voxel grid. On the other hand, they train separate networks to process different combinations of front/side/top views, while our updater network allows us to fuse information from any of the 13 viewpoints available in our interface. In addition, we integrated our approach in an interactive system to demonstrate the novel workflow it enables.

3 OVERVIEW

Figure 2 provides an overview of our system and the underlying convolutional neural networks. The left part of the figure illustrates the offline training of the deep neural networks. Given a dataset of 3D models, we first generate a voxel representation of each object, along with a series of line drawings rendered from different viewpoints. Our single-view CNN takes a drawing as input and generates a voxel grid with probabilistic occupancy. Our updater CNN also takes a drawing as input, and complements it with an initial 3D reconstruction provided by the single view network. Note that we transform this reconstruction according to the camera matrix of the input drawing, so that the updater CNN does not have to learn the mapping between the 3D volume and a given viewpoint. The updater network fuses the information from these two inputs to output a new 3D reconstruction. In practice, we repeatedly loop the updater over all available drawings of a shape to converge towards a multi-view coherent solution.

The right part of Figure 2 illustrates our online modeling workflow. The main motivation of our approach is to provide a workflow that seamlessly combines 2D sketching and 3D visualization. At the beginning of a modeling session, our interface displays an empty 3D space seen from a 3/4 view. We additionally display perspective guidance to help users draw with the same perspective as the one used to generate the training data, as detailed in Section 6. Once an initial drawing is completed, the user can invoke our single-view CNN to obtain its 3D reconstruction, which she can visualize from any viewpoint. The user can then refine the shape by re-drawing it from a new viewpoint, using the current reconstruction as a reference. We feed each new drawing to the updater network to generate an improved 3D reconstruction.

4 VOLUMETRIC PREDICTION FROM LINE DRAWINGS

The key enabler of our modeling system is a deep convolutional network that we train to predict voxelized objects from line drawings. We first present our single-view network that takes as input one drawing to generate an initial 3D reconstruction. We then introduce our *updater network* that iteratively fuses multi-view information by taking as input a drawing and an existing volumetric prediction. We illustrate our network in Figure 3 and provide a detailed description in supplemental materials. We discuss and compare our design choices against alternative solutions in Section 7.

4.1 Single view prediction

Our single-view network follows an encoder-decoder architecture typical of image generation tasks such as depth prediction [Eigen and Fergus 2015], colorization [Sangkloy et al. 2017], novel view synthesis [Park et al. 2017]. The encoder passes the input image through a series of convolutions of stride 2 and rectified linear units to progressively reduce spatial resolution while increasing feature dimensionality, effectively extracting a compact representation of the image content. The decoder passes this representation through a series of deconvolutions of stride 2 and rectified linear units to progressively generate a new visualization of the image content, in our case in the form of a voxel grid.

Following [Ronneberger et al. 2015], we also include skip connections between the encoder and decoder layers of equal resolution. These skip connections allow local information to bypass the encoder bottleneck, providing the decoder with multi-scale features that capture both global context and fine image details. Isola et al. [2017] have demonstrated the effectiveness of a similar "U-net" architecture for image-to-image translation tasks.

The task of our network is to classify each voxel as occupied or empty. We model the voxel grid as a multi-channel image, where each channel corresponds to one depth slice. Given this representation, our network can be seen as an extension of existing depth prediction networks [Eigen and Fergus 2015], where we not only predict the depth of the visible surface but also all occluded voxels along the viewing ray corresponding to each pixel of the input drawing. Since our modeling interface employs a perspective camera model, the voxel grid associated to a drawing actually forms a pyramid in 3D space. While we considered using an orthographic camera model for simplicity, our early experiments suggest that perspective cues significantly help the network to predict depth for regular shapes.

4.2 Multi-view prediction

Our updater network adopts a similar architecture as the one described above, except that it also takes as input an existing volumetric prediction and uses the input drawing to refine it. In practice, we concatenate the existing prediction with the output of the second convolution layer, as illustrated in Figure 3 (yellow block). Note that we do not threshold the probabilities of occupancy in the existing prediction, which allows the updater network to account for the uncertainty of each voxel.

Iterative update. The updater network processes one drawing at a time, which allows us to handle an unbounded number of views. However, each update may modify the prediction in a way that is not coherent with the other views. We found that we can achieve multi-view consistency by iteratively applying the updater network until convergence, akin to a coordinate descent optimization. Figure 4 illustrates this process with two views: the first drawing is given as input to the single-view network to generate a first prediction. This prediction is then given to the updater network along with the second drawing to produce a refined solution. The resulting voxel grid can now be processed again by the updater, this time taking the first drawing as input. This process generalizes to more views by looping the updater over all drawings in sequence. In practice, we used 5 iterations for all results in the paper. We evaluate the convergence of this iterative scheme in Section 7.

Resampling the voxel grid. As mentioned in Section 4.1, we designed our networks to process and generate voxel grids that are

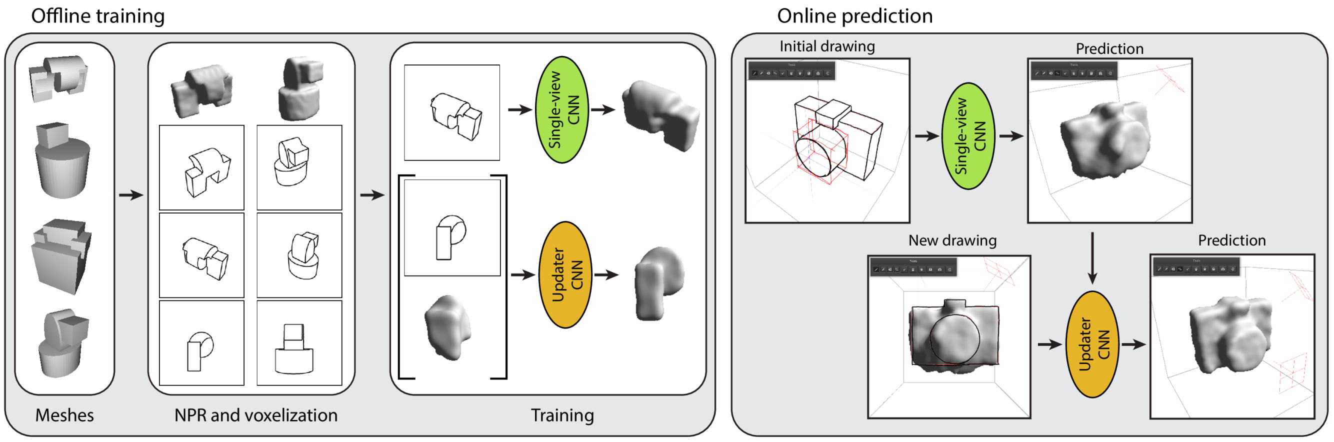


Figure 2: Overview of our method. Left: We train our system with a large collection of 3D models, from which we generate voxel grids and synthetic drawings. We train a single-view CNN to predict an initial reconstruction from a single drawing, as well as an updater CNN to refine a reconstruction given a new drawing. Right: The single-view CNN allows users to obtain a complete 3D shape from a single drawing. Users can refine this initial result by drawing the shape from additional viewpoints. The updater CNN combines all the available drawings to generate the final output.

expressed in the coordinate system of the input drawing. When dealing with multiple drawings, the prediction obtained with any drawing needs to be transformed and resampled to be passed through the updater network with another drawing. In practice, we store the prediction in a reference voxel grid in world coordinates, and transform this grid to and from the coordinate system of each drawing on which we run the updater network.

Single-view refinement. While we designed the updater network to fuse information between multiple views, we found that it is also able to refine a single-view prediction when used as a feedback loop on a single drawing, as shown in Figure 5. This observation may seem counter-intuitive, since the updater does not have more information than the single-view network in that configuration. We hypothesize that iterating the updater on a single drawing emulates a deeper network with higher capacity. Note also that a similar iterative refinement has been demonstrated in the context of pose estimation [Carreira et al. 2016; Oberweger et al. 2015].

5 DATA GENERATION AND TRAINING

The two CNNs outlined above require pairs of drawings and ground truth voxel grids for training. Ideally, the training data should be representative of the distribution of drawings and 3D shapes that users of our system are likely to create. However, while datasets of cartoon drawings of objects have been collected via crowd-sourcing [Eitz et al. 2012b], building a dataset of perspective drawings registered with 3D models raises many additional challenges. In particular, we assume that users of our system are sufficiently skilled to draw 3D objects in approximate perspective, which may not be the case for the average Mechanical Turk worker [Eitz et al. 2012a]. In addition, crowd-sourcing such a drawing task would require significant time and money, which prevents iterative design of the dataset, for instance to adjust the complexity and diversity of the 3D shapes.

Similarly to recent work on sketch-based procedural modeling [Huang et al. 2016; Nishida et al. 2016], we bypass the challenges of real-world data collection by generating our training data using non-photorealistic rendering. This approach gives us full control over the variability of the dataset in terms of shapes, rendering styles, and viewpoints.

5.1 3D objects

The main strength of our data-driven approach is its ability to capture the characteristic features of a class of objects. We experimented with two sources of 3D object datasets: online repositories and shape grammars.

Online repositories. A first usage scenario of our system is to train it with specific object classes. For instance, a furniture designer could train the system with chairs and tables, while a car designer could train the system with various vehicles. In practice, we tested this approach with the two largest classes of the ShapeCOSEG dataset [Wang et al. 2012], which contain 400 chairs and 300 vases, some of which are shown in Figure 6. For each dataset, we used 90% of the objects for training and the other 10% for testing.

Shape grammars. One drawback of online shape repositories is that they are dominated by a few object classes, such as tables, chairs, cars and airplanes [Chang et al. 2015], which may not cover the types of objects the user wants to model. In addition, many of these objects are very detailed, while we would like our system to also handle simple shapes to allow coarse-to-fine explorative design. We address these limitations by training our system with abstract shapes with varying degrees of complexity.

We designed a simple procedure that generates shapes by combining cubes and cylinders with CSG operations. Our procedure iteratively constructs a shape by adding or substracting random primitives. At each iteration, we position the new primitive on a

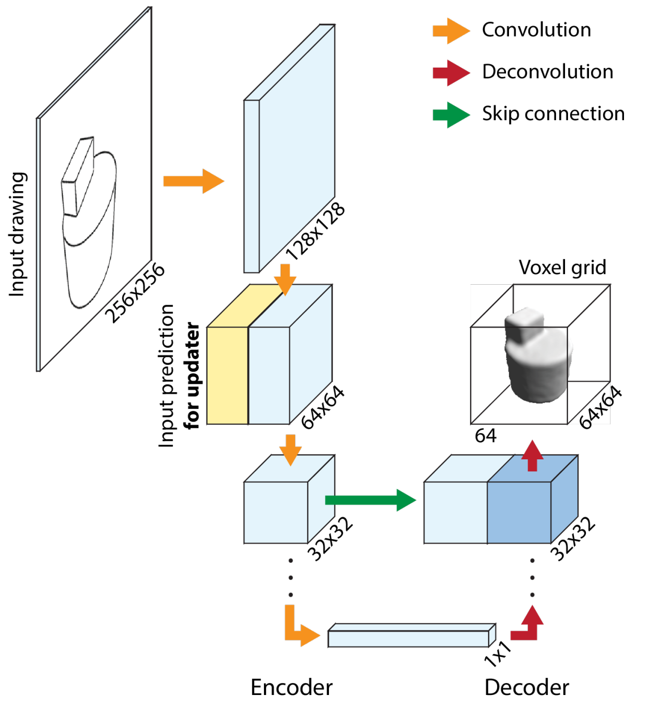


Figure 3: Our network follows a so-called "U-net" encoder-decoder architecture. The input drawing is processed by a series of convolution and rectified linear units to extract high-dimensional features at low spatial resolution. These features are then processed by deconvolutions and rectified linear units to generate the multi-channel image that represents our voxel grid. Skip connections, shown in green, concatenate the output of convolutional layers to the output of deconvolutional layers of the same resolution. These connections allow high-resolution features to bypass the encoder bottleneck, allowing the network to exploit multi-scale information for decoding. The updater network also takes an existing prediction as input, shown in yellow.

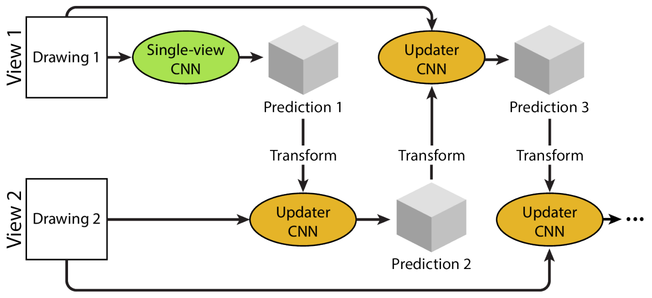


Figure 4: We apply the updater network iteratively, alternating between views to converge towards a multi-view coherent solution. Here we illustrate a few iterations between two views, although we loop over more views when available.

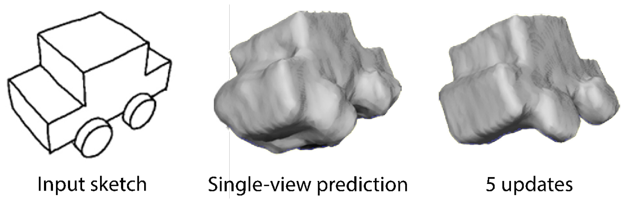


Figure 5: The updater network can refine the prediction even when only one drawing is available.

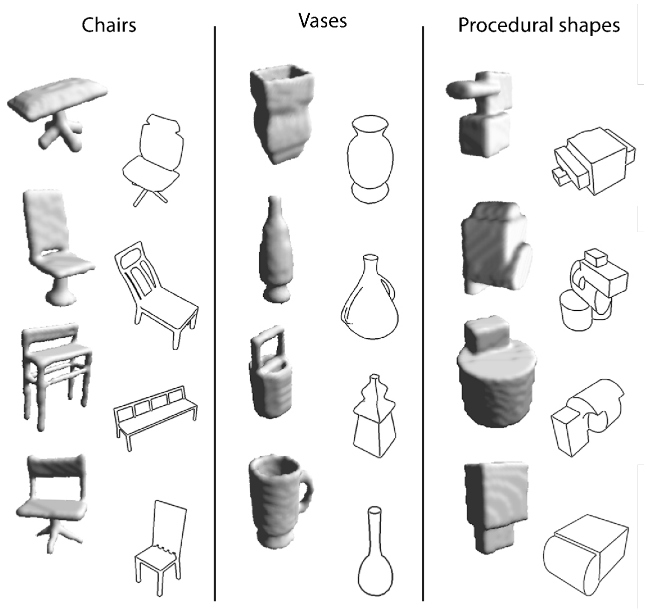


Figure 6: Representative voxelized objects and drawings from our three datasets.

random face of the existing shape, scale it by a random factor in each dimension and displace it by a small random vector while maintaining contact. The primitive is either merged with or subtracted from the existing shape. We inject high-level priors in this procedure by aligning each primitive with one of the three world axes, and by symmetrizing the shape with respect to the xy plane in world coordinates. The resulting axis-aligned, symmetric shapes resemble man-made objects dominated by flat orthogonal faces, yet also contain holes, concavities and curved parts. We generated $20,000$ random shapes with this procedure, some of which are shown in Figure 6. We isolated 50 of these shapes for testing, and used the rest for training.

Voxelization. We voxelize each object at a resolution of 64^3 voxels using Binvox [Min 2016; Nooruddin and Turk 2003]. We scale each object so that the voxel grid covers 120% of the largest side of the object's bounding box.

5.2 Line rendering

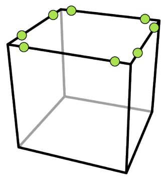
We adopt the image-space contour rendering approach of Saito and Takahashi [1990], who apply an edge detector over the normal and depth maps of the object rendered from a given viewpoint. Edges in the depth map correspond to depth discontinuities, while edges in the normal map correspond to sharp ridges and valleys. We render each drawing at a resolution of 256^2 pixels.

An exciting direction of future work would be to train our system with other rendering techniques such as suggestive contours [DeCarlo et al. 2003] and hatching [Hertzmann and Zorin 2000] to cover a wider range of styles.

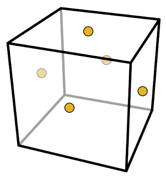
5.3 Viewpoints

Viewpoint is a major source of ambiguity for line drawing interpretation. We now describe our strategies to significantly reduce ambiguity for the single-view network by restricting camera orientation and position. We relax these restrictions for the updater network since it can handle more ambiguity thanks to the existing prediction it also takes as input.

Camera orientation. Representing a 3D object with a single drawing necessarily induces ambiguity. The design literature [Eissen and Steur 2011] as well as other sketching systems [Bae et al. 2008; Shao et al. 2012; Xu et al. 2014] recommend the use of "informative" perspective viewpoints that reduce ambiguity by showing the 3D object with minimal foreshortening on all sides. We follow this practice to train our single-view network. We render each object from eight viewpoints positioned near the top corners of its bounding box, as shown in inset.



In addition, designers frequently adopt so-called "accidental" viewpoints when representing a shape with several drawings, such as the common front, side and top views. We include these viewpoints in the training set of our updater network since we found them useful to refine axis-aligned shapes. However, we do not use these viewpoints with the single-view network because they often yield significant occlusions, which make them very challenging to interpret in the absence of additional information. The inset shows the additional viewpoints available to the updater network.



Camera position. Line drawings also have an inherent depth ambiguity: the same drawing can represent a small object close to the camera, or a big object far from the camera. We reduce such ambiguity for the single-view network by positioning the 3D object at a constant distance to the camera. In addition, we achieve invariance to 2D translations in the image plane by displacing the camera by a random vector perpendicular to the view direction.

However, a 2D translation in one view potentially corresponds to a translation in depth in another view, which prevents us imposing a constant distance to the camera for the updater network. We thus train the updater network with random 3D displacements of the camera. We found that the updater network succeeds in exploiting the existing prediction to position the object in depth.

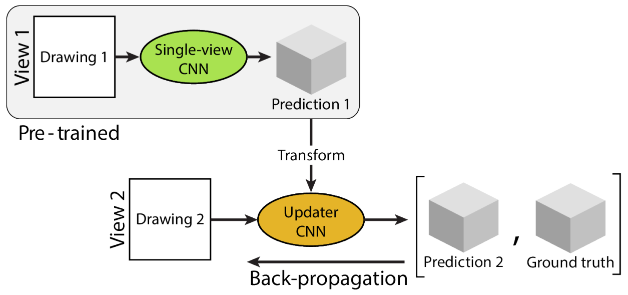


Figure 7: We first train our single-view network on ground truth data, then use its predictions as training data for the updater network.

5.4 Training procedure

We train our single view network by providing a line drawing as input and a ground truth voxel grid as output. However, training our updater network is more involved since we also need to provide an existing prediction as input. Given a drawing and its associated 3D model, we obtain an initial prediction by running the single-view network on another viewpoint of the same object. Figure 7 illustrates this process. We thus need to train the single-view network before training the updater.

We trained our system using the Adam solver [Kingma and Ba 2014], using batch normalization [Ioffe and Szegedy 2015] to accelerate training. We fixed Adam's parameters to $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e-8$. We fixed the learning rate to 0.0002 and trained the networks for 1, 000, 000 iterations. Training the complete system took around a week on a NVidia TitanX GPU.

6 USER INTERFACE

Figure 8 shows the interactive interface that we built around our deep 3D reconstruction engine. We designed this interface to reproduce traditional pen-on-paper freehand drawing. However, we introduced several key features to guide users in producing drawings that match the characteristics of our training data in terms of viewpoints and perspective.

Similarly to the seminal Teddy system [Igarashi et al. 1999], the working space serves both as a canvas to draw a shape and as a 3D viewer to visualize the reconstruction from different viewpoints. While we allow free viewpoint rotations for visualization, we restrict the drawing viewpoints to the ones used for training. In particular, we impose a 3/4 view for the first drawing, and snap the camera to one of the 13 viewpoints available to the updater for subsequent drawings.

The menu in the top left allows users to switch from 2D drawing to 3D navigation and also provides basic drawing tools (pen and eraser). In addition, we provide a "construction line" mode to draw scaffolds [Schmidt et al. 2009] and other guidance that will not be sent to the network (shown in red in our interface). We found such lines especially useful to lay down the main structure of the object before drawing precise contours (shown in black). We further facilitate perspective drawing by displaying three orthogonal vanishing

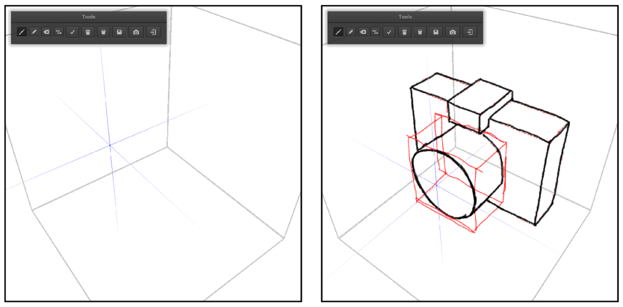


Figure 8: Screenshots of our user interface. We display axis-aligned lines around the cursor to guide perspective drawing (left, shown in blue). We also allow users to draw construction lines (right, shown in red). Only the black lines are processed by our 3D reconstruction engine.

lines centered on the pen cursor (shown in blue) and by delineating the working space with a wireframe cube.

For each voxel, our networks estimate the probability that it is occupied. We render the shape by ray-casting the 0.5 iso-surface of this volume, using the volumetric gradient to compute normals for shading. We also export the shape as a triangle mesh, which we obtain by apply a marching cube [Lorensen and Cline 1987] followed by a bilateral filter to remove aliasing [Jones et al. 2003].

7 RESULTS AND EVALUATION

We now evaluate the expressivity and robustness of our method and compare it to alternative approaches. We use the dataset of abstract procedural shapes for these comparisons, and separately evaluate the impact of the other datasets on our method. All results were obtained with a voxel grid of resolution 64^3.

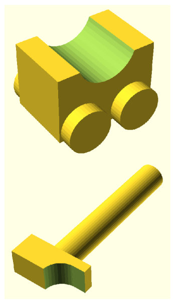
In all cases we evaluate the quality of a volumetric reconstruction against ground-truth using the intersection-over-union (IoU) metric, which computes the ratio between the intersection and the union of the two shapes [Häne et al. 2017; Riegler et al. 2017]. The main advantage of this metric over the classification accuracy is that it ignores the many correctly-classified empty voxels far away from the shapes.

7.1 Creative modeling by experts

Figure 9 presents several 3D scenes modeled with our system by two expert users. These results were created with the version trained on abstract procedural shapes, which succeeds in interpreting these drawings of diverse man-made shapes. In particular, the CNNs manage to segment the foreground object from its background, combine information from different drawings to reconstruct occluded parts, create holes and concavities such as on the armchairs and on the last wagon of the train. Figure 10 shows the more challenging case of a house with a slanted roof, which is well reconstructed even though the networks were only trained with shapes made of axis-aligned cuboids and cylinders.

We provide screen captures of a few modeling sessions in the accompanying video, showing how users iterate between 2D sketching and 3D navigation within a single workspace. In particular, users can draw a complete shape from one viewpoint before rotating the 3D model to continue working on it from another viewpoint. This workflow contrasts with the one of existing sketching systems that require users to decompose the object in simple parts [Nishida et al. 2016] or to provide multiple drawings of the shape before obtaining its reconstruction [Rivers et al. 2010]. The accompanying video also shows 3D visualizations of the objects, and the supplementary materials contain the corresponding 3D mesh files.

7.2 Evaluation by novice users



While we designed our system for artists who know how to draw in perspective, we also conducted a small study to evaluate whether our interface is usable by novices. We recruited six participants with limited drawing and 3D modeling skills (average score of 2.8 and 2.3 respectively on a 5-point Likert scale from 1 = *poor* to 5 = *good*). All participants followed a 15 minutes tutorial to learn how to draw a cube and a cylinder within our interface. We then asked each participant to model one of the two objects shown in inset, which we designed to be expressible by our shape grammar. Figure 11 shows the drawings and 3D models they created.

Overall, participants quickly managed to use our system (average score of 5.5 on a 7-point Likert scale from 1 = *hard to use* to 7 = *easy to use*). However, many participants were disappointed by the lack of details of the reconstruction and gave an average score of 3.8 when asked if the 3D model corresponds well to their drawings (1 = *not at all*, 7 = *very well*). The best results were obtained by participants who planned their drawings ahead to best represent the shape centered on screen (P1 and P6). In contrast, two participants did not obtain a complete shape because they drew the object too small to capture details (P2) or too big to fit in the drawing area (P5). This observation suggests the need for additional guidance to help novices compose a well-proportioned perspective drawing.

All participants judged the on-cursor vanishing lines helpful to draw in perspective (6.6 on average on a 7-point Likert scale from 1 = *not helpful* to 7 = *very helpful*). P3 commented *"Sometimes it seems to me that the guides point to wrong directions, but that is just my sense of perspective that is wrong!"*. All the participants followed the vanishing lines to draw cuboid shapes. However, several participants commented that they would have liked guidance to draw 3D cylinders. In particular, P2 drew very approximate cylinders to represent the wheels of his car, which our system failed to interpret properly.

Finally, even though P1 and P6 created many drawings, several are redundant and did not help our system improve its prediction. We believe that users would interact more effectively with our system if we could indicate which regions of the shape is under-constrained. Recent work on uncertainty quantification in deep networks form a promising direction to tackle this challenge [Kendall and Gal 2017].

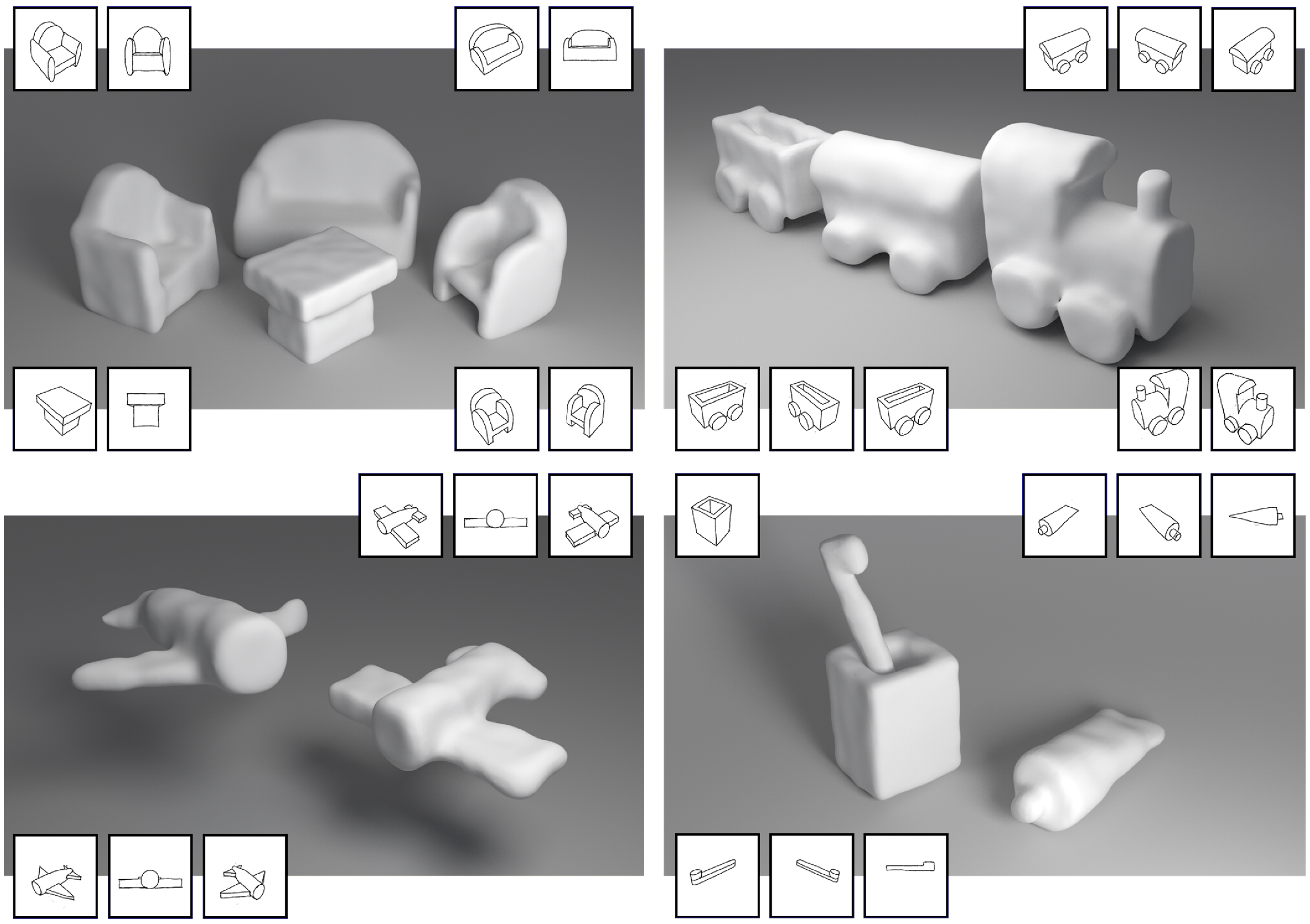


Figure 9: 3D scenes modeled using our system. Each object was modeled with two to three hand drawings, shown in insets.

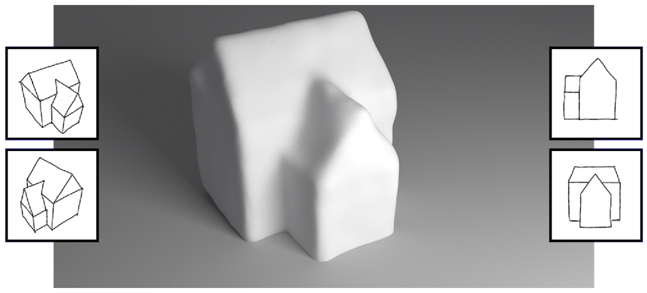


Figure 10: Our system manages to reconstruct the slanted roof of this house, even though it was only trained on shapes composed from axis-aligned cuboids and cylinders.

7.3 Datasets

One of the motivations for our deep-learning-based approach is to allow adaptation to different classes of objects. Figure 13 provides a quantitative evaluation of this ability for the single-view network. This figure plots reconstruction quality over the three testing datasets, using all three training datasets. As expected, the network trained on a given dataset performs best on this dataset, showing its specialization to a specific class. For instance, only the network trained on vases succeeds to create hollow shape from an ambiguous drawing (fourth row, third and fourth column). Interestingly, the network trained on abstract procedural shapes is second best on the other datasets, suggesting a higher potential for generalization. Figure 12 shows representative results for each condition. While the networks trained on chairs and vases manage to reconstruct objects from these classes, they fail to generalize to other shapes. In contrast, the network trained on abstract shapes captures the overall shape of chairs and vases, although it misses some of the details. This superiority of the procedural dataset may be due to its larger size and variability.

7.4 Convergence of the updater

In what follows, we count one iteration each time the updater network visits all views in sequence. Figure 14(left) plots the L^2

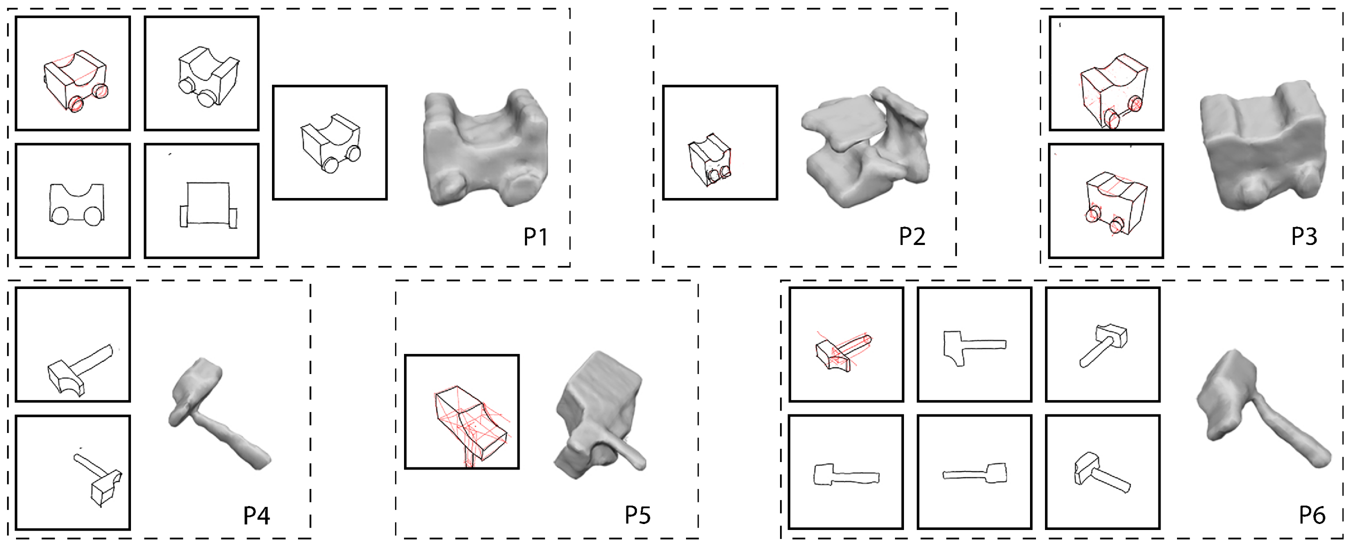


Figure 11: Drawings and 3D objects created by our six novice participants. P1 and P6 obtained the best results by drawing the object in the center of the canvas, with proper perspective. In contrast, P2 drew the object too small and with too approximate perspective to be reconstructed by our system, while P5 left too little room for the handle of the hammer.

distance between successive iterations, averaged over 50 abstract shapes rendered from two, three and four random views. While we have no formal proof of convergence, this experiment shows that the algorithm quickly stabilizes to a unique solution. However, Figure 14(right) shows that the accuracy decreases slightly with iterations. We suspect that this loss of accuracy is due to the fact that the updater is only trained on the output of the single-view network, not on its own output. However, training the updater recursively would be more involved. We found that 5 iterations provide a good trade-off between multi-view coherence and accuracy.

7.5 Robustness

While we trained our networks with clean drawings rendered with perfect perspective, they offer robustness to moderate sources of noise, such as wavy, incomplete or overshot lines and slight perspective distortions, as shown in Figure 15. However, drawings made under drastically different or wrong perspectives yield distorted shapes. We also observed sensitivity to over-sketching and varying line thickness. An interesting direction for future work would be to render the training data using advanced non-photorealistic rendering, in the hope of achieving invariance to line style.

7.6 Comparisons

To the best of our knowledge, our method is the first that can automatically reconstruct a 3D model from a set of multiple perspective bitmap drawings. As a baseline, we compare our approach with a silhouette carving algorithm [Martin and Aggarwal 1983]. We implemented two versions of silhouette carving for this comparison. The first version takes as input the same drawings as the ones provided to our method, which necessarily includes a 3/4 view for the first drawing to be fed to the single-view network, and different random views for the other drawings. The second version only takes drawings from orthogonal views, which is the most informative setup for silhouette carving. As shown in Figure 17, our approach outperforms silhouette carving in both conditions. In particular, our method achieves a high IoU ratio with as little as one view. Figure 16 provides a visual comparison between our reconstructions and the ones by silhouette carving. Our approach is especially beneficial in the presence of concavities.

Figure 18 evaluates our network architecture against several alternative designs. We perform this evaluation on the single-view network since any improvement made on it would directly benefit the updater. A first important design choice to evaluate is the choice of the volumetric representation. While we chose a binary representation of the volume, we also considered a signed distance function. However, our experiments reveal that this alternative representation reduces quality slightly, producing smoother predictions than ours. We also compare our U-net architecture with the multi-scale depth prediction network proposed by Eigen and Fergus [2015], which we modified to output a multi-channel image. This network follows a similar encoder-decoder strategy as ours but does not include as many skip-connections between multi-scale layers, which also reduces the quality of the prediction.

7.7 Limitations

Figure 19 shows drawings with thin structures that are challenging to reconstruct for our current implementation based on a 64^3 voxel grid. High-resolution volumetric representations is an active topic in deep learning [Fan et al. 2017; Häne et al. 2017; Riegler et al. 2017] and we hope to benefit from progress in that field in the near future. An alternative approach is to predict multi-view depth maps, as proposed by Lun et al. [2017], although these depth maps need to be registered and fused by an optimization method to produce the final 3D surface.

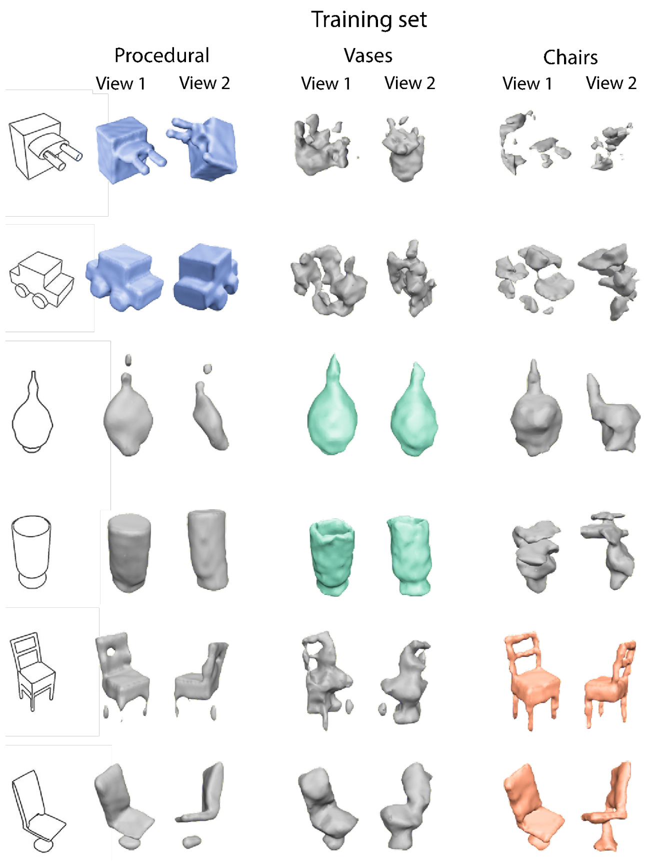


Figure 12: We trained the single-view network with three different datasets, and evaluated each version on the three testing sets. Each version performs best on the testing set for which it was trained, showing that the network learns to specialize to specific object categories. The network performs better on abstract procedural shapes than on chairs and vases, which contain more thin structures and for which the training set is smaller.

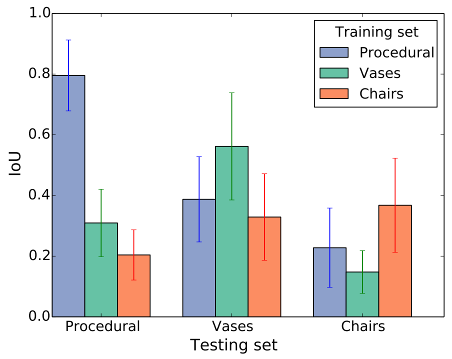


Figure 13: We trained the single-view network with three different datasets (depicted with different colors), and evaluated each version on drawings from the three testing sets (distributed on the x-axis). The network trained on abstract procedural shapes captures the overall shape of objects from other categories, while the networks trained on chairs and vases generalize poorly. Each network performs best on the shapes for which it has been trained.

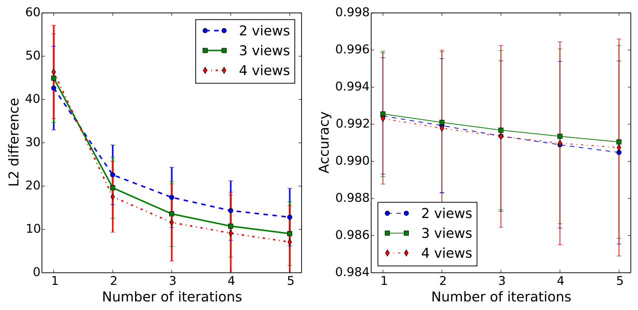


Figure 14: Left: Difference of prediction between successive iterations of the updater network, showing that the network quickly converges towards a stable solution. Right: The accuracy decreases slightly during the iterations. 5 iterations offer a good trade-off between multi-view coherence and accuracy.

Our deep networks also have difficulty interpreting drawings with many occlusions, as shown in Figure 20. Fortunately, designers tend to avoid viewpoints with many occlusions since they are not the most informative. Nevertheless, occlusions are inevitable on objects composed of many parts, and we observed that the quality of the reconstruction can reduce as users add more details to their drawings. A simple solution to this limitation would consist in letting the user freeze the reconstruction before adding novel parts. This feature could be implemented by copying the reconstruction in a temporary buffer, and flagging all the lines as construction lines to be ignored by the system. Users could then proceed with drawing new parts which would be interpreted as a new object, and we could display the existing reconstruction and the new parts together by taking the union of their volumes.

7.8 Performances

We implemented our system using the Caffe library for deep learning [Jia et al. 2014] and OpenGL for real-time rendering in the user interface. Table 1 provides timings at test time for an increasing number of views, measured on a desktop computer with an NVidia TitanX GPU, and on a MacBook Pro laptop using only the CPU. Our 3D reconstruction engine scales linearly with the number of views and outputs a prediction in less than a second using GPU and within a few seconds using CPU, on a 64^3 voxel grid with 5 iterations of the updater. Our single-view and updater networks occupy 775MB of memory together.

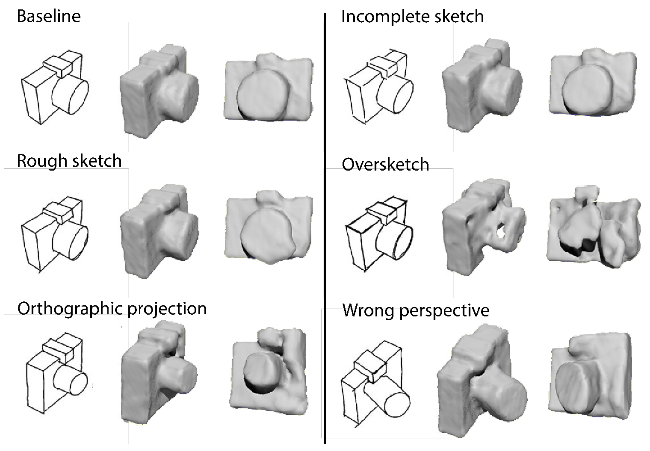


Figure 15: Behavior of the single-view network on various sources of noise. While the network trained on clean drawing tolerates some amount of sketchiness, overshoot and incompleteness, it is sensitive to over-sketching that produces thicker lines than the ones in the training set. Drawing with a very different or wrong perspective yields distorted shapes.

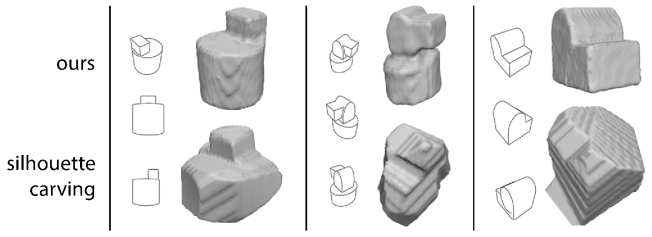


Figure 16: Reconstructed objects using our method (top row) and silhouette carving (bottom row) with 3 random views. Silhouette carving struggles to recover concavities.

	1 view	2 views	3 views	4 views
Desktop GPU (ms)	140	210	280	350
Laptop CPU (s)	1	1.5	2.2	2.9

Table 1: Our method scales linearly with the number of input drawings, generating the prediction in less than a second for a 64^3 voxel grid on a modern GPU.

8 DISCUSSION

Research in sketch-based modeling has long been driven by the need for a flexible method capable of reconstructing a large variety of shapes from drawings with minimal user indications. In this paper we explored the use of deep learning to reach this goal and proposed an architecture capable of predicting 3D volumes from a single drawing, as well as fusing information from multiple drawings via iterative updates. This architecture fits naturally in a simple modeling interface allowing users to seamlessly sketch and visualize shapes in 3D. Our approach is modular and we see multiple directions of future research to improve it.

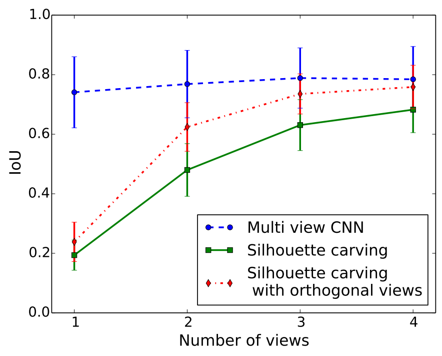


Figure 17: Comparison between our method (blue) and silhouette carving (green and red). The strength of our approach is that it achieves high accuracy from only one view, and remains competitive with silhouette carving with four views. In addition, our method can handle concavities that cannot be recovered by carving.

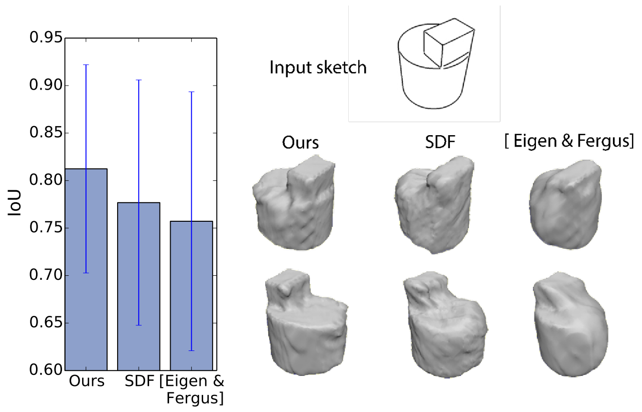


Figure 18: We compare our single-view network with the one of Eigen and Fergus [Eigen and Fergus 2015] and with a network trained to predict a signed-distance function rather than a binary voxel grid. Our design outperforms these two alternatives.

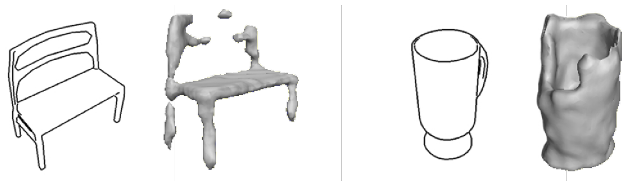


Figure 19: Thin structures are challenging to capture by the 64^3 voxel grid.

We demonstrated the potential of our system by training it with simple contour drawings. Artists often use other visual cues to

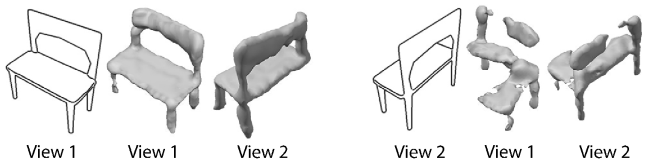


Figure 20: The single-view network performs best on informative viewpoints that minimize occlusions (left). Drawing the same shape from a viewpoint with significant occlusions results in an erroneous prediction (right).

depict shape in their drawings, such as hatching to convey shading [Hertzmann and Zorin 2000], cross-sections to convey curvature directions [Shao et al. 2012], scaffolds and vanishing lines to lay down perspective and bounding volumes [Schmidt et al. 2009]. An exciting direction of research would be to train our system to generalize to all these drawing techniques. However, achieving this goal may require the design of new non-photorealistic rendering algorithms that formalize and reproduce such techniques [Gori et al. 2017]. Going further, style transfer algorithms [Kalogerakis et al. 2012] may even enable the synthesis of user-specific training data.

Despite its simplicity, our abstract shape grammar proved sufficient to train our system to reconstruct a variety of man-made objects. We hope that this new application will motivate further research in the design of advanced shape grammars that capture the statistics of real-world objects.

We used simple thresholding to extract a 3D surface from the predicted occupancy grid. More advanced surface extraction algorithms such as graphcut segmentation [Boykov and Jolly 2001] could be used to further regularize the solution, for instance by incorporating a prior on piecewise-smooth surfaces. Alternatively, finer reconstructions may be obtained by training our CNNs with different loss functions. In particular, *adversarial networks* have recently shown an impressive ability to hallucinate fine details in synthesis tasks by combining a generator network with a discriminator that learns to identify if an output is real or synthesized [Isola et al. 2017].

In this work, we explored deep learning as an alternative to hand-crafted geometric optimizations. Fundamentaly, we see these two approaches as complementary and would like to use our predictions to initialize precise constraint-based optimizations that would strictly enforce regularities such as parallelism, orthogonality and symmetry [Li et al. 2011; Xu et al. 2014].

Finally, while we developed our iterative updater architecture to reconstruct objects from drawings, a similar architecture could be used for multiview 3D reconstruction from photographs given calibrated cameras. The challenge of such an approach is to obtain a sufficiently large amount of training data that covers not only different shapes, but also different textures, materials and lighting conditions as encountered in realistic scenes.

ACKNOWLEDGMENTS

Many thanks to Yulia Gryaditskaya for sketching several of our results, and for her help on the renderings and video. This work was supported in part by the ERC starting grant D^3 (ERC-2016-STG 714221), the Intel/NSF VEC award IIS-1539099, FBF grant 2018-0017, ANR project EnHerit (ANR-17-CE23-0008), research and software donations from Adobe, and by hardware donations from NVIDIA.

REFERENCES

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *ACM symposium on User Interface Software and Technology (UIST)*. 151–160. 2, 3, 7

H. Barrow and J. Tenenbaum. 1981. Interpreting Line Drawings as Three-Dimensional Surfaces. *Artificial Intelligence* 17 (1981). 2

Y. Y. Boykov and M. P. Jolly. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *IEEE International Conference on Computer Vision (ICCV)*, Vol. 1. 13

Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. 2016. Human pose estimation with iterative error feedback. (2016). 3, 5

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository*. Technical Report. 5

Qifeng Chen and Vladlen Koltun. 2017. Photographic Image Synthesis with Cascaded Refinement Networks. In *IEEE International Conference on Computer Vision (ICCV)*. 2, 3

Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 2013. 3-Sweep: Extracting Editable Objects from a Single Photo. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32, 6 (2013). 3

Christopher Bongsoo 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 *European Conference on Computer Vision (ECCV)*. 628–644. 3

Forrester Cole, Phillip Isola, William T Freeman, Frédo Durand, and Edward H Adelson. 2012. Shapecollage: occlusion-aware, example-based shape interpretation. In *European Conference on Computer Vision (ECCV)*. Springer, 665–678. 3

Frederic Cordier, Hyewon Seo, Mahmoud Melkemi, and Nickolas S. Sapidis. 2013. Inferring Mirror Symmetric 3D Shapes from Sketches. *Computer Aided Design* 45, 2 (Feb. 2013), 301–311. 2

Frederic Cordier, Karan Singh, Yotam Gingold, and Marie-Paule Cani. 2016. Sketch-based Modeling. In *SIGGRAPH ASIA Courses*. ACM, Article 18, 222 pages. 2

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive Contours for Conveying Shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22, 3 (July 2003), 848–855. 7

A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox. 2016. Learning to Generate Chairs, Tables and Cars with Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2016). 3

David Eigen and Rob Fergus. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision (ICCV)*. 2650–2658. 3, 4, 10, 12

Koos Eissen and Roselien Steur. 2011. *Sketching: The Basics*. Bis Publishers. 7

Mathias Eitz, James Hays, and Marc Alexa. 2012a. How Do Humans Sketch Objects? *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10. 5

Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. 2012b. Sketch-Based Shape Retrieval. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 31:1–31:10. 2, 3, 5

Haoqiang Fan, Hao Su, and Leonidas Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2017). 2, 3, 10

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: a Global Approach to Line Drawing Vectorization. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2016). 2

Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. 2003. A Search Engine for 3D Models. *ACM Transactions on Graphics* 22, 1 (Jan. 2003), 83–105. 3

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

Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. FlowRep: Descriptive Curve Networks for Free-Form Design Shapes. *ACM Transaction on Graphics (Proc. SIGGRAPH)* 36, 4 (2017). 13

X. Han, C. Gao, and Y. Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36, 4 (July 2017). 3

Christian Häne, Shubham Tulsiani, and Jitendra Malik. 2017. Hierarchical Surface Prediction for 3D Object Reconstruction. In *International Conference on 3D Vision (3DV)*. 8, 10

Aaron Hertzmann and Denis Zorin. 2000. Illustrating smooth surfaces. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 517–526. 7, 13

Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. 2016. Shape Synthesis from Sketches via Procedural Models and Convolutional Networks. *IEEE*

Transactions on Visualization and Computer Graphics (TVCG) 22, 10 (2016), 1. 2, 3, 5

Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Transactions on Graphics* (2015). 3

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: A Sketching Interface for 3D Freeform Design. In *SIGGRAPH*. 409–416. 2, 3, 7

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of The 32nd International Conference on Machine Learning*. 448–456. 7

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2017). 4, 13, 15

Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. 2017. SurfaceNet: An End-To-End 3D Neural Network for Multiview Stereopsis. In *IEEE International Conference on Computer Vision (ICCV)*. 3

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678. 11

Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. 2003. Non-Iterative, Feature-Preserving Mesh Smoothing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2003). 8

Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. 2015. Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. *ACM Transactions on Graphics* (2015). 2

Evangelos Kalogerakis, Derek Nowrouzezahrai, Simon Breslav, and Aaron Hertzmann. 2012. Learning Hatching for Pen-and-Ink Illustration of Surfaces. *ACM Transactions on Graphics* 31, 1 (2012). 13

Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In *Advances in Neural Information Processing Systems 30 (NIPS)*. 8

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). 7

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*. 1097–1105. 3

Jeehyung Lee and Thomas Funkhouser. 2008. Sketch-Based Search and Composition of 3D Models. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (SBIM)*. 3

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36, 4 (2017). 3

Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. 2011. GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Transactions on Graphics (proc. SIGGRAPH)* 30, 4 (2011). 13

H Lipson and M Shpitalni. 1996. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design* 28, 8 (1996), 651–663. 2

H. Lipson and M. Shpitalni. 2000. Conceptual Design and Analysis by Sketching. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 14, 5 (Nov. 2000), 391–401. 3

J. Liu, F. Yu, and T. Funkhouser. 2017. Interactive 3D Modeling with a Generative Adversarial Network. In *International Conference on 3D Vision (3DV)*. 3

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2015). 3

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH* 21, 4 (Aug. 1987), 163–169. 8

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. In *International Conference on 3D Vision (3DV)*. 3, 10

J. Malik and D. Maydan. 1989. Recovering three-dimensional shape from a single image of curved objects. *IEEE Pattern Analysis and Machine Intelligence (PAMI)* 11, 6 (1989), 555–566. 2

Worthy N. Martin and J. K. Aggarwal. 1983. Volumetric Descriptions of Objects from Multiple Views. *IEEE Transactions on Pattern Analysis Machine Intelligence* 5, 2 (Feb. 1983), 150–158. 10

Patrick Min. 2016. Binvox 3D mesh voxelizer. http://www.google.com/search?q=binvox. (2016). Accessed: 2016-11-01. 6

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26, Article 41 (2007). Issue 3. 3

A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems (NIPS)*. 3

Gen Nishida, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive Sketching of Urban Procedural Models. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2016). 2, 3, 5, 8

F. S. Nooruddin and G. Turk. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 9, 2 (April 2003), 191–205. 6

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015. Training a feedback loop for hand pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*. 3316–3324. 3, 5

L. Olsen, F.F. Samavati, M.C. Sousa, and J. Jorge. 2009. Sketch-Based Modeling: A Survey. *Computers & Graphics* 33 (2009). Issue 1. 2

Gunay Orbay and Levent Burak Kara. 2012. Sketch-Based Surface Design Using Malleable Curve Networks. *Computers & Graphics* 36, 8 (2012), 916–929. 3

Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. 2017. Transformation-Grounded Image Generation Network for Novel 3D View Synthesis. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2017). 3, 4

Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. 2017. OctNet-Fusion: Learning Depth Fusion from Data. In *International Conference on 3D Vision (3DV)*. 8, 10

Alec Rivers, Frédo Durand, and Takeo Igarashi. 2010. 3D Modeling with Silhouettes. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29, 4, Article 109 (2010), 8 pages. 2, 3, 8

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*. 234–241. 4

Takafumi Saito and Tokiichiro Takahashi. 1990. Comprehensible Rendering of 3-D Shapes. *SIGGRAPH* 24, 4 (1990), 197–206. 7

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2016). 3

Patsorn Sangkloy, Jingwan Lu, Chen Fang, FIsher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2017). 3, 4

Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 149. 2, 7, 13

Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: shading concept sketches using cross-section curves. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31, 4 (2012). 2, 7, 13

Alex Shtof, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. 2013. Geosemantic Snapping for Sketch-Based Modeling. *Computer Graphics Forum* 32, 2 (2013), 245–253. 3

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35, 4 (2016). 3

Suraj Srinivas, Ravi Kiran Sarvadevabhatla, Konda Reddy Mopuri, Nikita Prabhu, Srinivas S. S. Kruthiventi, and R. Venkatesh Babu. 2016. A Taxonomy of Deep Convolutional Neural Nets for Computer Vision. Vol. 2. 36. 2

Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters. *ACM Transactions on Graphics* 33 (2014). 3

Fang Wang, Le Kang, and Yi Li. 2015b. Sketch-based 3d shape retrieval using convolutional neural networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 1875–1883. 3

Xiaolong Wang, David Fouhey, and Abhinav Gupta. 2015a. Designing deep networks for surface normal estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 539–547. 3

Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2012. Active Co-Analysis of a Set of Shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* (2012). Issue 4. 5

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 *Neural Information Processing Systems (NIPS)*. 3

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shape Modeling. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2015). 3

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. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 233–245. 2, 3

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014). 2, 7, 13

Qiuying Xu, Yotam Gingold, and Karan Singh. 2015. Inverse Toon Shading: Interactive Normal Field Modeling with Isophotes. In *Proceedings of Sketch-Based Interfaces and Modeling (SBIM)*. 2

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2Image: Conditional Image Generation from Visual Attributes. In *European Conference on Computer Vision (ECCV)*. Springer, 776–791. 3

Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. 1996. SKETCH: An Interface for Sketching 3D Scenes. In *SIGGRAPH*. ACM, 163–170. 1, 2

Youyi Zheng, Han Liu, Julie Dorsey, and Niloy Mitra. 2016. SMART CANVAS : Context-inferred Interpretation of Sketches for Preparatory Design Studies. *Computer Graphics Forum* 35, 2 (2016), 37–48. 3

APPENDIX

We adapt our architecture from [Isola et al. 2017] by reducing the number of layers of the decoder part.

Let (De)Ck denote a (De)Convolution-BatchNorm-ReLU layer with k filters (output has k channels). (De)CDk denotes a (De)Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All convolutions are 4ÃŮ4 spatial filters applied with stride 2. Convolutions in the encoder downsample by a factor of 2, whereas deconvolutions in the decoder upsample by a factor of 2.

The encoder-decoder architecture consists of:

$$\begin{tikzcd}
C64 \arrow[r] & C128 \arrow[r] & C256 \arrow[r] \arrow[d, dashed] & C512 \arrow[r] \arrow[d, dashed] & C512 \arrow[r] \arrow[d, dashed] & C512 \arrow[r] \arrow[d, dashed] & C512 \arrow[r] \arrow[d, dashed] & C512 \arrow[d] \\
 & & DeC128 & DeC256 \arrow[l] & DeC512 \arrow[l] & DeCD512 \arrow[l] & DeCD512 \arrow[l] & DeCD512 \arrow[l]
\end{tikzcd}$$

After the last layer of the decoder, a SoftMax is applied followed by a classification loss (multinomial logistic loss). We then keep only one channel over two (the one containing the probability of occupancy) to get a voxel grid of dimension 64.

As an exception to the above notation, Batch-Norm is not applied to the first C64 layer in the encoder. All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky.

The skip connections (shown as dashed arrows) consist in concatenating the output of a convolution in the encoder to a deconvolution of the same size in the decoder.