Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches

Benoit Guillard[*], Edoardo Remelli[*], Pierre Yvernay, Pascal Fua

CVLab, EPFL

name.surname@epfl.ch

Abstract

Reconstructing 3D shape from 2D sketches has long been an open problem because the sketches only provide very sparse and ambiguous information. In this paper, we use an encoder/decoder architecture for the sketch to mesh translation. When integrated into a user interface that provides camera parameters for the sketches, this enables us to leverage its latent parametrization to represent and refine a 3D mesh so that its projections match the external contours outlined in the sketch. We will show that this approach is easy to deploy, robust to style changes, and effective. Furthermore, it can be used for shape refinement given only single pen strokes.

We compare our approach to state-of-the-art methods on sketches—both hand-drawn and synthesized—and demonstrate that we outperform them.

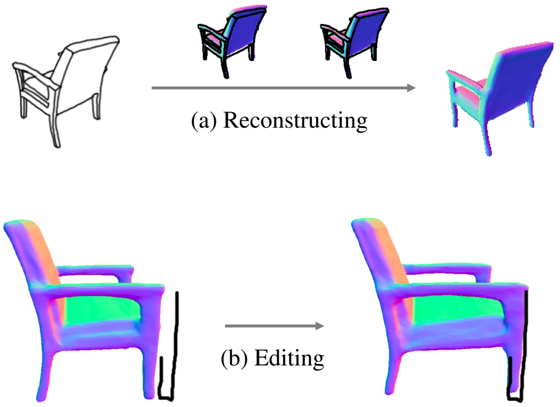


Figure 1. **Sketch2Mesh.** We propose a pipeline for reconstructing and editing 3D shapes from line drawings. We train an encoder/decoder architecture to regress surface meshes from synthetic sketches. Our network learns a compact representation of 3D shapes that is suitable for downstream optimization: **(a)** When presented with sketches drawn in a style different from that of the training ones– for example a real drawing – aligning the projected external contours to the input sketch bridges the domain gap. **(b)** The same formulation can be used to enable unexperienced users to edit reconstructed shapes via simple 2D pen strokes. Best seen in Supplemental video.

1. Introduction

Reconstructing 3D shapes from hand-drawn sketches has the potential to revolutionize the way designers, industrial engineers, and artists interact with Computer Aided Design (CAD) systems. Not only would it address the industrial need to digitize vast amounts of legacy models, an insurmountable task, but it would allow practitioners to interact with shapes by drawing in 2D, which is natural to them, instead of having to sculpt 3D shapes produced by cumbersome 3D scanners.

Current deep learning approaches [26, 6, 45, 46] that regress 3D point clouds and volumetric grids from 2D sketches have shown promise despite being trained on synthetic data, but yield coarse 3D surface representations that are cumbersome to edit. Furthermore, they require multi-view sketches for effective reconstruction [6] or are restricted to a fixed set of views [26].

Meanwhile Single View Reconstruction (SVR) approaches have progressed rapidly thanks to the introduction of new shape representations [11, 33, 30, 36] along with novel architectures [43, 10, 13, 44] that exploit image-plane feature pooling to align reconstructions to input images. Hence, it can seem like a natural idea to also use them for reconstruction from sketches. Unfortunately, as we will show, the sparse nature of sketch images makes it difficult for state-of-the-art SVR networks relying on local feature pooling from the image plane to perform well. This difficulty is compounded by the fact that different people sketch differently, which introduces a great deal of variability in the training process and makes generalization problematic. Furthermore, these architectures do not learn a compact representation of 3D shapes, which makes the learned models

[*] Equal contribution

unsuitable for down stream applications requiring a strong shape prior, such as shape editing.

To overcome these challenges, we train an encoder/decoder architecture [36] to produce a 3D mesh estimate given an input line drawing. This yields a compact latent representation that acts as an information bottleneck. At inference time, given a previously unseen camera-calibrated sketch, we compute the corresponding latent vector and refine its components to make the projected 3D shape it parameterizes match the sketch as well as possible. In effect, this compensates for the style difference between the input sketch and those that were used for training purposes. We propose and investigate two different ways to do this:

1. *Sketch2Mesh/Render*. We use a state-of-the-art image translation technique [17] trained to synthesize foreground/background images from sketches and then use the resulting images as targets for differentiable rasterization [38, 36, 34].

2. *Sketch2Mesh/Chamfer*. We directly optimize the position of the 3D shape's projected contours to make them coincide with those of the input sketch by minimizing a 2D Chamfer distance.

Remarkably, *Sketch2Mesh/Chamfer*, even though simpler, works as well or better than *Sketch2Mesh/Render*. The former exploits only *external* object contours for refinement purposes, which helps with generalization because most graphics designers draw these external contours in a similar way. It also makes it unnecessary the auxiliary network that turns sketches into foreground/background images.

A further strength of *Sketch2Mesh/Chamfer* is that it does not require backpropagation from a full rasterized image but only from sparse contours. Hence, it is naturally applicable for local refinement given a camera-calibrated partial sketch. And, unlike earlier work [32, 20, 21] on shape editing from local pen strokes it allows us to take into account a strong shape prior by relying on the latent vector, ensuring that shapes can be edited robustly with sparse 2D pen strokes.

2. Related Work

Recent years have seen an explosion in 3D shape modeling capabilities from images in general and sketches in particular. In this section, we first review some of the new shape representation methods that have made this possible and then discuss how specific advances relate to the method we propose.

Surface Representation. Among existing 3D surface representations, meshes made of vertices and faces are one of the most popular and versatile types and many early surface-modeling methods focused on deforming pre-existing templates based on such meshes that were either limited by design to a fixed topology [8, 40] or required *ad hoc* heuristics that do not generalize well [29]. Furthermore, because meshes can have variable numbers of vertices and facets, it is challenging to make this representation suitable to deep learning architectures. A standard approach has therefore been to use graph convolutions to deform a pre-defined template [31, 43]. Hence, it is limited to a fixed topology by design. A promising alternative [11] is to use a union of surface patches instead, which can handle arbitrary topologies. However, this method does not offer any guarantee that patches stitch together correctly and, in practice, yields non watertight surfaces.

Another alternative is to use an implicit description where the surface is described by the zero crossings of a volumetric function $\Psi : R^3 \rightarrow R$ [41] whose values can be adjusted. The strength of this implicit representation is that the zero-crossing surface can change topology without explicit re-parameterization. Until recently, its main drawback was thought to be that working with volumes, instead of surfaces, massively increased the computational burden.

This changed dramatically two years ago with the introduction of continuous deep implicit-fields. They represent 3D shapes as level sets of deep networks that map 3D coordinates to a signed distance function [33] or an occupancy field [30, 3]. This mapping yields a continuous shape representation that is lightweight but not limited in resolution.

However, for applications requiring explicit surface parameterizations, the non-differentiability of standard approaches to iso-surface extraction [25] remains an obstacle to exploiting the advantages of implicit representations. This was overcome recently by introducing a differentiable way to produce explicit surface mesh representations from Deep Signed Distance-Functions [36]. It was shown that, by reasoning about how implicit-field perturbations affect local surface geometry, one can differentiate the 3D location of surface samples with respect to the underlying deep implicit-field. This insight resulted in the *MeshSDF* end-to-end differentiable architecture that takes as input a compact latent vector and outputs a 3D watertight mesh and that we use here.

3D Reconstruction from Sketches. Reconstructing 3D models from line drawings has also been an active research area for more many decades. Early attempts tackled the inherent ambiguity of this inverse problem by either assuming that the drawn lines represent specific shape features [27, 15] or by constraining the class of 3D shapes that can be handled [22, 24, 4, 19]. More recently inflatable surface models [7] demonstrated easy animation of the reconstructed shapes, but still constrain the artist to draw from a side view of the object and are limited to a fixed topology. The emergence of deep learning has given rise to models

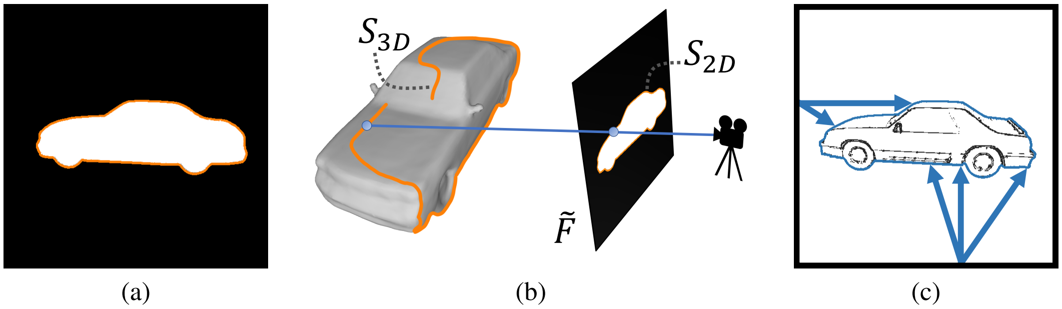


Figure 2. **External contours in 2D and 3D. (a)** The external contours of the projected mesh are shown in orange. They form the S_{2D} set of Eq. 4. **(b)** The corresponding 3D points on the mesh are also overlaid in orange. They form the S_{3D} set of Eq. 3. **(c)** We filter the original sketch to keep only the external contours, which will be matched against S_{2D}.

[26, 6, 18] that can be far more expressive and have therefore boosted both the performance and generalization of algorithms that parse sketches into 3D shapes. Given an input sketch, [26] regress depth and normal maps from 12 viewpoints, and fuse them to obtain a dense point cloud from which a surface mesh is extracted. Their pipeline, however, must be trained for each input sketch viewpoint, making it incompatible with a free viewpoint sketching interface. In [6], a 3D convolutional network trained on a catalog of simple shape primitives regresses occupancy grids from sketches. In addition to the limited output resolution, a refinement strategy based on sketches from multiple views is needed for effective reconstruction. [18] jointly projects 3D shapes and their front, side and top views occluding contours in the embedding space of a VAE. Their pipeline is trained on a single sketch style (occluding contours) and outputs volumetric grids. At inference time it retrieves the closest embedding code that was seen during training, thus limiting its generalization capabilities.

Single View Reconstruction. Recently, Single View Reconstruction (SVR) from RGB images has also experienced tremendous progresses thanks to both the introduction of new shape representations discussed above and to he introduction of new SVR architectures [43, 10, 13, 44] relying on image-plane feature pooling to align reconstructions to input images. Unfortunately, many of these methods rely on feature pooling and therefore lack a compact latent representation that can be used for downstream applications that require strong shape priors, such as refinement or editing. However, there are SVR methods that feature compact surface representations and we discuss below those that leverage either differentiable rendering or contours, as we do.

Refinement using differentiable rendering. Recent work [38, 36, 34] has shown that 2D buffers -such as silhouettes or depth maps- can be used to refine 3D reconstructions produced by encoder/decoder architectures and thus allow networks trained on synthetic RGB renders to yield accurate reconstruction on real world images. These approaches rely on either estimating 2D buffers from input images — using state-of-the-art segmentation/depth estimation networks trained on large-scale real world datasets [23] — or acquiring the additional information through specific sensors. Applying refinement techniques to line-drawings would require to use an auxiliary network to infer occupancy masks from input sketches. However we found that such networks struggle at generalizing to different sketching styles. This is due to the lack of diversified large-scale line-drawings datasets [12], and makes refinement through differentiable rasterization less effective, or in some cases detrimental.

Refinement by matching Silhouettes. Silhouettes have long been used to track articulated and rigid objects by modeling them using volumetric primitives whose occluding contours can be computed given a pose estimate. The quality of these contours can then be evaluated using either the chamfer distance to image edges [9] or more sophisticated measures [42, 1]. Other approaches to exploiting external contours rely on minimizing the distance between the 3D model and the lines of sights defined by these contours [16]. Our approach follows this tradition but combines silhouette alignment to a far more powerful latent representation.

3. Method

3.1. Formalization

Let $\mathbf{C} \in \{0,1\}^{H \times W}$ be a binary image representing a sketch and let $\Lambda : \mathbb{R}^3 \to \mathbb{R}^2$ denote the function that projects 3D points into that image. By convention, $\mathbf{C}[i,j]$ is 0 if it is marked by a pen stroke, and $\mathbf{C}[i,j] = 1$ otherwise.

We learn an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ such that $\mathcal{D} \circ \mathcal{E}(\mathbf{C})$ yields a mesh $\mathcal{M}_\Theta = (\mathbf{V}_\Theta, \mathbf{F}_\Theta)$. $\Theta = \mathcal{E}(\mathbf{C})$ is the latent vector that parameterizes our shapes. $\mathbf{V}_\Theta$ and $\mathbf{F}_\Theta$ represent the 3D vertices and facets. In practice, we use the MeshSDF encoding/decoding network architecture of [36]. In general, $\mathcal{M}_\Theta$ represents a 3D shape whose projection

$\Lambda(\mathcal{M}_\Theta)$ only roughly matches the sketch $\mathbf{C}$. Hence, our subsequent goal is to refine Θ so as to improve the match.

We can achieve this in of two ways. We can turn the sketch into a foreground/background image and use differentiable rasterization to ensure that the projection of $\mathcal{M}_\Theta$ matches that image. Alternatively, we can minimize the 2D Chamfer distance between the sketch and the projection. We describe both alternatives below.

3.2. Using Differential Rendering

In this method that we dubbed *Sketch2Mesh/Render*, we train an image translation technique [17] to synthesize foreground/background images from sketches. We denote as $\mathbf{M} \in \{0,1\}^{H \times W}$ this foreground/background image estimated from the input sketch $\mathbf{C}$. On the other hand, we use the differentiable rasterizer [35] $\mathcal{R}^{F/B}$ to render a foreground/background mask $\widetilde{\mathbf{M}} = \mathcal{R}_\Lambda^{F/B}(\mathcal{M}_\Theta)$ of the projection of $\mathcal{M}_\Theta$ by Λ. In $\widetilde{\mathbf{M}}$, a pixel value is 1 if it projects to the surface of the mesh $\mathcal{M}_\Theta$, and 0 otherwise. Finally, we refine $\mathcal{M}_\Theta$ shape by minimizing

$$\mathcal{L}_{F/B} = \left\| \mathbf{M} - \widetilde{\mathbf{M}} \right\|^2 \ , \tag{1}$$

the L_2 difference between $\mathbf{M}$ and $\widetilde{\mathbf{M}}$ with respect to Θ.

While conceptually straightforward, this approach is in fact quite complex because it depends on two off-the-shelf but complex pieces of software, the rasterizer [35] and image-translator [17], one of which has to be trained properly. We now turn to a simpler technique that can be implemented from scratch and does not rely on an auxiliary neural network.

3.3. Minimizing the 2D Chamfer Distance

The simpler *Sketch2Mesh/Chamfer* approach involves directly finding those 3D mesh points that project to the contour of the foreground image and then minimizing the Chamfer distance between this contour and the sketch.

3.3.1 Finding External Contours in 2D and 3D

To identify surface points on $\mathcal{M}_\Theta$ that project to exterior contour pixels, we first use Λ to project the whole mesh onto a $H \times W$ binary image $\widetilde{\mathbf{F}}$ in which all pixels are one except those belonging to external contours, such as those shown in orange in Fig. 2(a). Then, for each zero-valued pixel $\mathbf{p}$ in $\widetilde{\mathbf{F}}$, we look for a 3D point $\mathbf{P}$ on one of the mesh facets that projects to it, that is, a point that is visible and such that $\Lambda(\mathbf{P}) = \mathbf{p}$. In theory, this can be done by finding to which facet $\mathbf{p}$ belongs and then computing the intersection between the line of sight and the plane defined by that facet. In practice, we use Pytorch3d [35] which provides us with the facet number along with the barycentric coordinates of $\mathbf{P}$ within that facet. Hence, we write

$$\mathbf{P} = \alpha_1 \mathbf{V}_1 + \alpha_2 \mathbf{V}_2 + \alpha_3 \mathbf{V}_3 \ , \tag{2}$$

with $\mathbf{V}_1$, $\mathbf{V}_1$ and $\mathbf{V}_3$ are the vertices of the fact to which $\mathbf{P}$ belongs and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Since the coordinates of the three vertices are differentiable functions of Θ, so are those of $\mathbf{P}$. Repeating this operation for all external contour points yields a set of 3D points S_{3D} such that

$$\forall \mathbf{P} \in S_{3D} \quad \widetilde{\mathbf{F}}[\Lambda(\mathbf{P})] = 0 \ , \tag{3}$$

along with a corresponding set of 2D projections

$$S_{2D} = \{\Lambda(\mathbf{P}) | \mathbf{P} \in S_{3D}\} \ . \tag{4}$$

Fig. 2(b) depicts such a set.

3.3.2 Objective function

To exploit the target sketch $\mathbf{C}$, we first filter it to only preserve external contours. To this end, we shoot rays from the 4 image borders and only retain the first black pixels hit by a ray, as shown in Fig. 2(c). This yields a filtered sketch $\mathbf{F} \in \{0,1\}^{H \times W}$. As before, $\mathbf{F}[\mathbf{p}] = 0$ for pixels $\mathbf{p}$ belonging to external contour and $\mathbf{F}[\mathbf{p}] = 1$ for others. The ray-shooting algorithm we use is described in details in the supplementary material.

Our goal being for $\mathbf{F}$, the filtered sketch, and $\widetilde{\mathbf{F}}$, the external contours of the projected triangulation introduced in Section 3.3.1, to match as well as possible, we write the objective function to be minimized as the bidirectional 2D Chamfer loss

$$\mathcal{L}_{CD} = \sum_{\mathbf{u} \in S_{2D}} \min_{\mathbf{v} | \mathbf{F}[\mathbf{v}]=0} \|\mathbf{u} - \mathbf{v}\|^2 + \sum_{\mathbf{v} | \mathbf{F}[\mathbf{v}]=0} \min_{\mathbf{u} \in S_{2D}} \|\mathbf{u} - \mathbf{v}\|^2 \ . \tag{5}$$

The coordinates of the 3D vertices in S_{3D} are differentiable with respect to Θ. Since Λ is differentiable, so are their 2D projections in S_{2D} and $\mathcal{L}_{CD}$ as whole.

3.4. Using a Partial Sketch

Minimizing the 2D Chamfer distance between external contours as described above does not require the input sketch to depict the shape in its entirety. This enables us to take advantage of partial sketches made of a single stroke. In this case, we can simply take the filtered sketch $\mathbf{F}$ introduced above to be the sketch itself. But we must ensure that parts of the surface which project far away from the sketch remain unchanged. The rationale for this is that the initial shape should be preserved except where modifications are specified. To this end, we regularize the refinement procedure as follows.

Given the initial value Θ_0 of the latent vector we want to refine along with differentiable rasterizers [35] $\mathcal{R}^N$ and $\mathcal{R}^{F/B}$ that return the normal maps $\mathbf{N}_\Theta$ and foreground/background mask $\mathbf{M}_\Theta$ given mesh $\mathcal{M}_\Theta$, respectively, we minimize

$$\begin{aligned} \mathcal{L}_{partial} = {} & \mathcal{L}_{CD} \\ & + \|\mathbb{1}_t \circ (\mathbf{M}_\Theta - \mathbf{M}_{\Theta_0})\|^2 + \|\mathbb{1}_t \circ (\mathbf{N}_\Theta - \mathbf{N}_{\Theta_0})\|^2 \end{aligned} \tag{6}$$

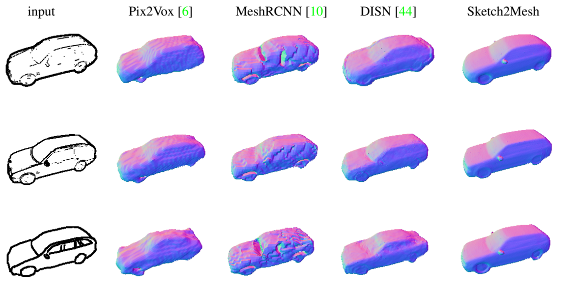


Figure 3. **Robustness to changes in sketch style.** Given a `Suggestive` sketch (top), a `SketchFD` one (middle), or a hand-drawn one (bottom), Sketch2Mesh—unlike Pix2Vox, MeshRCNN, and DISN—yields reconstructions that are similar to each other and close to the ground-truth.

where $\mathcal{L}_{CD}$ is the Chamfer distance of Eq. 5, $\mathbb{1}_t$ is a mask that is zero within a distance t of the sketch and one further away, and $\circ$ is the element wise product. In other words, the parts of the surface that project near to the sketch should match it and the others should keep their original normals and boundaries.

Crucially, this is something that could not be done using the approach of Section 3.2, which requires complete sketches. This comes at the cost of having to use a differential renderer, unlike the approach of Section 3.3. But this still does not require a trained network for image translation, which makes it easy to deploy.

4. Results

4.1. Datasets

Publicly available large-scale line-drawings datasets with associated 3D models are rare. We therefore test our approach on two datasets, one for chairs that is available [46] and another for cars that we created ourselves. To further test, and crucially, to train our approach, we used 3D models from the well-established ShapeNet [2] to render 2D sketches.

Rendered Car and Chair Sketches. We use the car and chair categories from ShapeNet [2] both for training and testing. We adopt the same train/test splits as in [36]. For cars we use 1311 training samples and 113 test samples. The equivalent numbers are 5268 and 127 for chairs. For each object and corresponding 3D mesh, we randomly sample 16 azimuth and elevation angles. The cameras point at the object centroid while their distance to it and their focal lengths are kept fixed. To demonstrate robustness to sketching style, we generate two different 256 × 256 binary sketches for each viewpoint, as shown in the top two rows a Fig. 3. We will refer to them as `Suggestive` and `SketchFD` sketches, as described below.

Suggestive. We use the companion software of [5] to render sketches displaying that contain both occluding and suggestive contours. Suggestive contours are lines drawn on visible parts of the surface where a true occluding contour would first appear given a minimal viewpoint change. They are designed to emulate real line drawings in which lines other lines than the occluding contours are drawn to increase expressivity.

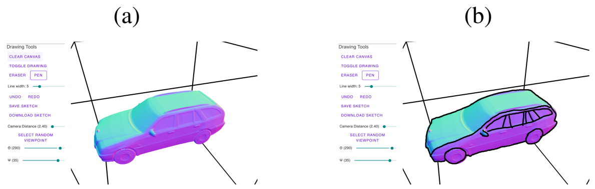


Figure 4. **Data acquisition interface.** (a) To guide unexperienced users and limit imprecision, we display the normal map as seen from a specific viewpoint. (b) The user can use a pen to draw freely on the resulting image.

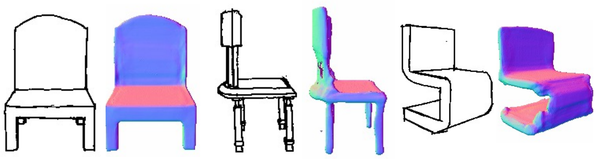


Figure 5. **ProSketch.** Input hand-drawn chair sketches and *Sketch2Mesh* reconstructions.

SketchFD. We also use the older rendering approach of [39]. We run an edge detector over the normal and depth maps of the rendered object. Edges in the depth map correspond to depth discontinuities while edges in the normal map correspond to sharp ridges and valleys. This yields synthetic sketches that, although conveying the same information, look very different from the ones on [5], as can be seen on the left of Fig. 3.

Hand-Drawn Car Sketches We asked 5 students with no prior experience in 3D design to draw by-hand the 113 cars from the ShapeNet test set. To this end, we developed the sketching interface depicted by Fig. 4 that runs on a standard tablet. The participants drew over normal maps rendered from the selected viewpoint so as to provide them guidance and ensure they all drew a similar car and used a known perspective. However, they were free to make the pen strokes they wanted. Hence, this dataset thus exhibits natural variations of style. To allow for comparison with results on the rendered sketches, we used the same viewpoint, which we will use to demonstrate that style change by itself is an obstacle to generalization for many methods.

Hand-Drawn Chair Sketches We use 177 chair sketches from the Prosketch dataset [12]. The chairs are seen from the front, profile, or a 45°azimuth view, as shown in Fig. 5. These viewpoints do not match the randomly selected ones we used for training, which makes this dataset especially challenging. Sample sketches and reconstructions are shown in Fig. 5

4.2. Metrics

As reconstruction metric, we use a 3D Chamfer loss (CD-l_2, the lower the better). It is computed by sampling $N = 10000$ points on the reconstructed mesh to form a first point cloud $\mathbf{C}_1$ and N on the ground truth mesh to form a second point cloud $\mathbf{C}_2$. We then compute

$$\text{CD-}l_2 = \frac{1}{N}\sum_{x\in\mathbf{C}_1}\min_{y\in\mathbf{y}}\|x-y\|^2 + \frac{1}{N}\sum_{y\in\mathbf{C}_2}\min_{x\in\mathbf{x}}\|y-x\|^2 \ .$$

We also report a normal consistency measure (NC, the higher the better), by taking the average pixel-wise dot product between normal maps of the reconstructed shape and the ground truth one.

4.3. Choosing the Best Method

Recall from the method section, that we have proposed two variants of our approach to refining our 3D meshes. *Sketch2Mesh/Render* operates by turning the sketch into a foreground/background image and minimizing the distance between that image and the mesh projection while *Sketch2Mesh/Chamfer* deforms the mesh to minimize the 2D Chamfer distance between the external contours of its projection and those of the sketch.

Once the latent representation has been learned on either `Suggestive` or `SketchFD` contours, *Sketch2Mesh/Chamfer* can be used without any further training. By contrast, *Sketch2Mesh/Render* requires an image translation network to predict foreground/background masks from sketches. Here, we use the one of [17] with a UNet [37] as its generator and in the LSGAN setting [28]. We train four separate instances of it on ShapeNet , one for each shape category (cars and chairs) and for each sketch rendering style (`Suggestive` and `SketchFD`).

This being done, we can compare *Sketch2Mesh/Render* against *Sketch2Mesh/Chamfer* on the test sets for both categories of object and the three categories of drawing we use, `Suggestive`, `SketchFD`, and `Hand Drawn`. We show qualitative results in Figs. 5 and 11. We report quantitative results in Tab. 1 for models trained on `Suggestive` contours. Similar results on `SketchFD` contours are presented in the supplementary material. Overall, both *Sketch2Mesh/Render* and *Sketch2Mesh/Chamfer* improve the initial metrics but *Sketch2Mesh/Chamfer* appears to be more robust to style changes. In other words, *Sketch2Mesh/Render* overfits to the style it is trained on and does not do as well as *Sketch2Mesh/Chamfer* when tested on a different one. Adding this to the fact, that *Sketch2Mesh/Chamfer*, unlike *Sketch2Mesh/Render*, does not require to train an auxiliary network clearly makes it the better approach. We will therefore use it in the remainder of the paper except otherwise noted and will refer to it as *Sketch2Mesh* for brevity.

4.4. Comparison against State-of-the-Art Methods

We now compare *Sketch2Mesh* against state-of-the-art methods that produce watertight meshes as we do. To this end, we train the architecture of [6] that regresses volumetric grids from sketches, which we dub *Pix2Vox*. We also compare to recent SVR method that rely on perceptual feature pooling from the image plane DISN [44] and MeshRCNN [10]. For a fair comparison, we use them in conjunction with the same image encoder as we do, ResNet18 [14].

We show qualitative results in Fig. 3. We report quantitative results on ShapeNet Cars and Chairs in Tables 2 and 3 when the latent representation have been learned either on `Suggestive` or `SketchFD` contours. On cars, *Sketch2Mesh* clearly outperforms the other methods. On chairs, MeshRCNN is very competitive, especially in terms of CD-l_2. But, as shown in Fig. 7, the meshes it produces are hardly usable, even though we uses the *Pretty* setup of the algorithm that attempts to regularize them. This is a well known phenomenon reported by its authors themselves. By contrast, our meshes can directly be used for downstream applications, without further preprocessing.

For completeness, we note that a very recent paper [45] also advocates using foreground/background masks to improve 3D reconstruction from sketches. However, instead of refining the mesh produced by a network using such as a mesh as done by *Sketch2Mesh/Render*, it recommends feeding the mask as an additional input to the network that produces the initial 3D shape. In Tab. 4, we compare this approach to ours when the network is trained using the `SketchFD` sketches on cars and tested on `Suggestive`. Both *Sketch2Mesh/Render* and *Sketch2Mesh/Chamfer* outperform it.

4.5. Interactive 3D editing

An important feature of *Sketch2Mesh* is that is can exploit sketches made of a single stroke to refine previously obtained shapes as discussed in Section 3.4, as shown in Fig. 8. To showcase the interactivity of our approach we built a web based user interface. The user may draw a sketch with the mouse or a touch enabled device and submit it to *Sketch2Mesh*. Then, successive partial sketches can also be input and matched by the optimizer. A video is provided in the supplementary material to show it in action.

5. Conclusion

We have proposed an approach to deriving 3D shapes from sketches that relies on an encoder/decoder architecture to compute a latent surface representation of the sketch.

Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l_2 \cdot 10^3 \downarrow$	*Initial*	1.613	4.658	6.818
	Sketch2Mesh/Render	**1.400**	4.253	5.752
	Sketch2Mesh/Chamfer	1.420	**3.132**	**4.395**
Normal Consistency ↑	*Initial*	91.14	84.73	81.40
	Sketch2Mesh/Render	**92.41**	86.18	83.88
	Sketch2Mesh/Chamfer	92.20	**87.00**	**84.75**

Cars

Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l_2 \cdot 10^3 \downarrow$	*Initial*	8.572	15.691	18.752
	Sketch2Mesh/Render	7.471	12.865	17.519
	Sketch2Mesh/Chamfer	**7.180**	**12.248**	**13.787**
Normal Consistency ↑	*Initial*	80.86	72.83	61.17
	Sketch2Mesh/Render	**83.99**	75.37	65.23
	Sketch2Mesh/Chamfer	82.61	**76.27**	**67.67**

Chairs

Table 1. **Cars and Chairs.** Reconstruction metrics when using the encoding/decoding network trained on `Suggestive` synthetic sketches of cars and of chairs, and tested on all 3 datasets. We show *initial* results before refinement and then using our two refinement methods. Note that *Sketch2Mesh/Chamfer* does better than *Sketch2Mesh/Render* on the styles it has *not* been trained for, indicating a greater robustness to style changes.

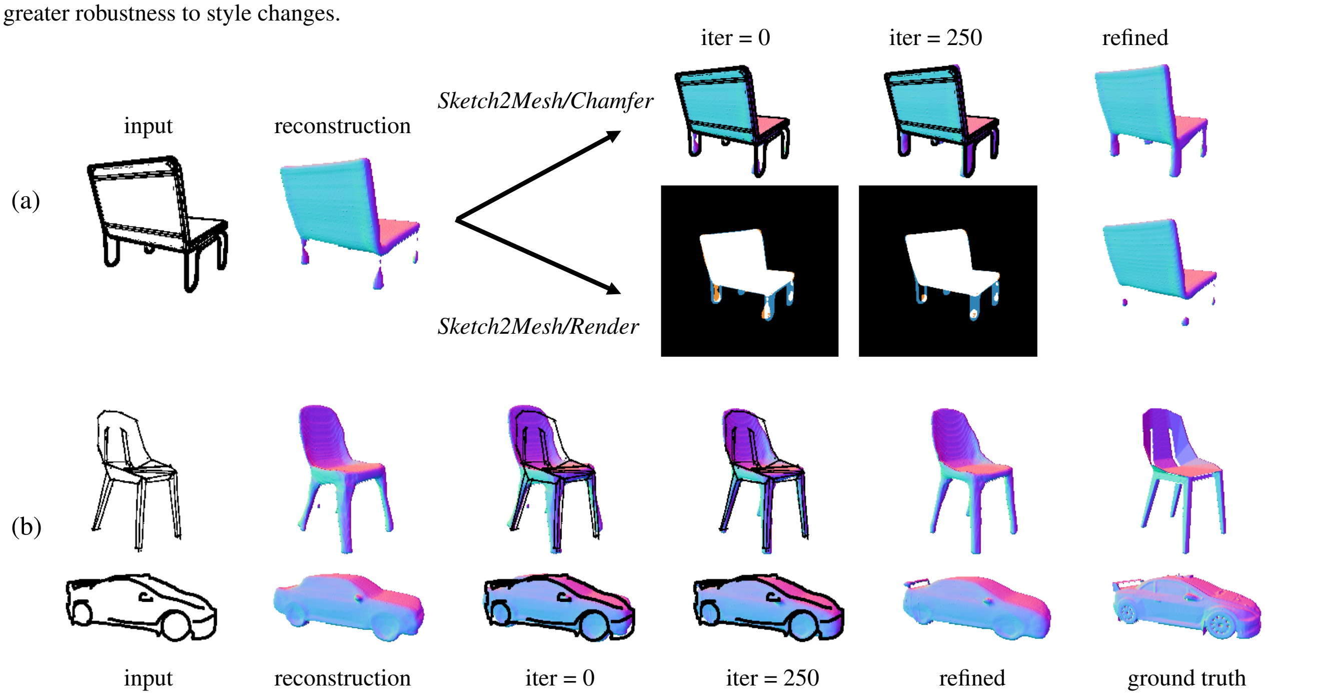


Figure 6. **Mesh refinement.** (a) Comparison of *Sketch2Mesh/Chamfer* (top) and *Sketch2Mesh/Render* (bottom). *Sketch2Mesh/Chamfer* handles thin components such as the legs of the chair better because it leverages sparse information. We examine this in more detail in the Supplementary material. (b) *Sketch2Mesh/Chamfer* results on challenging line drawings of a chairs and a car. We show the iterations from the initial mesh produced by the network that takes the sketch as input, which is then progressively refined.

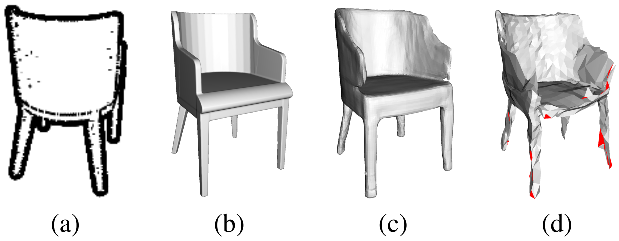
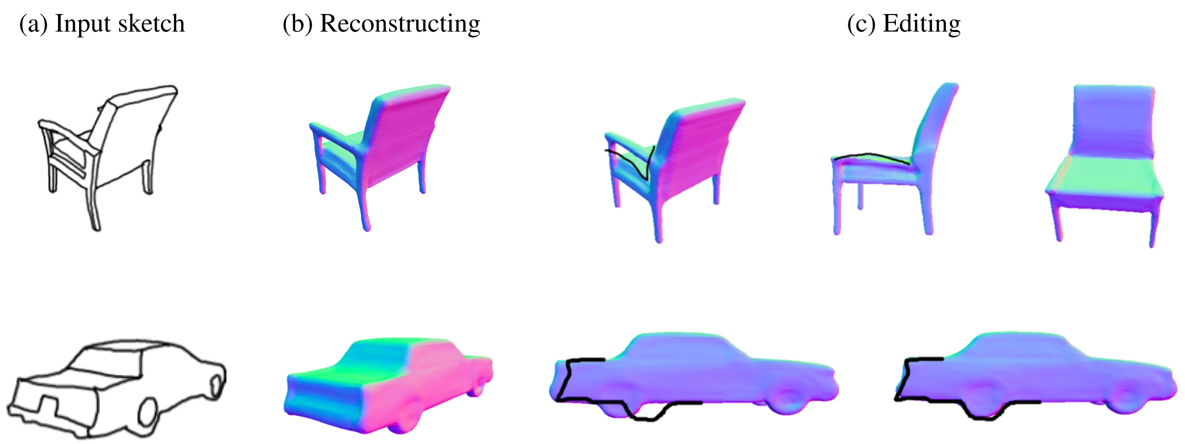


Figure 7. **Comparison with MeshRCNN: (a)** Input sketch **(b)** Ground truth shape, **(c)** *Sketch2Mesh* reconstruction, with CD-l_2=1.98, **(d)** MeshRCNN reconstruction, with CD-l_2=1.91. The flipped facets are shown in red. Despite having a slightly higher CD-l_2, our reconstruction is far more usable for further processing and, arguably, resembles the ground truth more than the MeshRCNN one.

It can in turn be refined to match the external contours of the sketch. It handles sketches drawn in a style it was not specifically trained for and outperforms state-of-the-art methods. Furthermore, it allows for interactive refinements by specifying partial 2D contours the object's projection must match, provided that perspective camera parameters are associated to the sketch. This can be achieved easily on a tablet using a stylus-based interface to draw.

We can see two natural improvements to our work. One is linked to the learned priors in our parametrization. Although the priors are usually good at preserving global shape properties such as symmetry, they can be either too constraining or not enough when for partial refinements. We would like some priors to actually be constraints—the wheels of the cars must be round and cannot touch the wheel wells and the feet of the chairs must all have the



(a) Input sketch (b) Reconstructing (c) Editing

Figure 8. **Interactive reconstruction & editing.** We developed an interface where the user can draw an initial sketch (a) to obtain its 3D reconstruction (b). It can then manipulate the object in 3D and draw one or more desired modifications (c). 3D surfaces are then optimized to match each constraint, solving the optimization problem of Section 3.4. The strong prior learned by our model allows to preserve global properties such as symmetry despite users provide sparse 2D strokes in input. Best seen in supplemental video.

Table 2. **Comparative results on Cars.**

Training Drawing Style: Suggestive				
Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l^2 \cdot 10^3 \downarrow$	Pix2Vox [6]	2.336	6.237	8.599
	MeshRCNN [10]	3.491	6.923	7.849
	DISN [44]	1.529	7.764	10.396
	Sketch2Mesh	**1.420**	**3.132**	**4.396**
Normal Consistency ↑	Pix2Vox [6]	89.07	80.49	76.70
	MeshRCNN [10]	84.19	79.93	77.91
	DISN [44]	92.15	79.51	72.52
	Sketch2Mesh	**92.20**	**87.00**	**84.74**
Training Drawing Style: SketchFD				
CD-$l^2 \cdot 10^3 \downarrow$	Pix2Vox [11]	3.529	2.475	3.146
	MeshRCNN [10]	3.117	3.596	4.829
	DISN [44]	4.036	1.573	3.763
	Sketch2Mesh	**2.419**	**1.516**	**2.047**
Normal Consistency ↑	Pix2Vox [11]	87.11	89.21	86.27
	MeshRCNN [10]	83.22	82.81	80.83
	DISN [44]	86.34	91.30	87.66
	Sketch2Mesh	**91.23**	**92.09**	**91.03**

Table 3. **Comparative results on Chairs.**

Training Drawing Style: Suggestive				
Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l^2 \cdot 10^3 \downarrow$	Pix2Vox [6]	22.953	33.46	62.132
	MeshRCNN [10]	**6.775**	**10.718**	19.055
	DISN [44]	7.045	18.104	23.282
	Sketch2Mesh	7.180	12.248	**13.787**
Normal Consistency ↑	Pix2Vox [11]	73.01	64.28	40.12
	MeshRCNN [10]	76.91	72.77	58.03
	DISN [44]	80.44	54.10	51.81
	Sketch2Mesh	**82.61**	**76.27**	**67.67**
Training Drawing Style: SketchFD				
CD-$l^2 \cdot 10^3 \downarrow$	Pix2Vox [11]	34.759	22.690	46.687
	MeshRCNN [10]	**9.530**	**5.812**	16.620
	DISN [44]	13.059	8.628	18.104
	Sketch2Mesh	**9.524**	6.737	**12.585**
Normal Consistency ↑	Pix2Vox [11]	65.97	72.52	52.96
	MeshRCNN [10]	77.62	**84.75**	69.76
	DISN [44]	73.39	80.21	62.58
	Sketch2Mesh	**81.00**	83.10	**70.39**

Table 4. **Comparison with the approach of [45].**

Method	Metric	
	CD-$l^2 \cdot 10^3 \downarrow$	NC ↑
MeshSDF [36]	3.231	89.67
MeshSDF [36] + mask	3.124	90.05
Sketch2Mesh/Render	2.538	90.92
Sketch2Mesh/Chamfer	**2.419**	**91.23**

same length, for example—in addition to those imposed by 2D sketches so that our technique can be turned into a full-fledged tool for Computer Assisted Design.Another research direction would be to incorporate interior lines in our refinement process. This is also an interesting challenge since we don't want to sacrifice the generalization ability this simple technique allowed us to achieve.

6. Acknowledgments

This work was supported in part by the Swiss National Science Foundation and by the Swiss Innovation Agency.

References

[1] A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *Conference on Computer Vision and Pattern Recognition*, 2004. 3

[2] A. Chang, T. Funkhouser, L. G., P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*, 2015. 5

[3] Z. Chen and H. Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[4] Frederic Cordier, Hyewon Seo, Mahmoud Melkemi, and

Nickolas S Sapidis. Inferring mirror symmetric 3d shapes from sketches. *Computer-Aided Design*, 45(2):301–311, 2013. 2

[5] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive Contours for Conveying Shape. *ACM SIGGRAPH*, 22(3):848–855, July 2003. 5

[6] J. Delanoy, M. Aubry, P. Isola, A. Efros, and A. Bousseau. 3D Sketching using Multi-View Deep Volumetric Prediction. *ACM on Computer Graphics and Interactive Techniques*, 1(1):1–22, 2018. 1, 3, 5, 6, 8

[7] M. Dvorožňák, D. Sỳkora, C. Curtis, B. Curless, O. Sorkine-Hornung, and D. Salesin. Monster mash: a single-view approach to casual 3D modeling and animation. *ACM Transactions on Graphics*, 2020. 2

[8] P. Fua. Model-Based Optimization: Accurate and Consistent Site Modeling. In *International Society for Photogrammetry and Remote Sensing*, July 1996. 2

[9] D.M. Gavrila and L.S. Davis. 3D Model-Based Tracking of Human Upper Body Movement: A Multi-View Approach. In *IEEE International Symposium on Computer Vision*, pages 253–258, November 1995. 3

[10] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In *International Conference on Computer Vision*, 2019. 1, 3, 5, 6, 8

[11] T. Groueix, M. Fisher, V. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 8

[12] Y. Gryaditskaya, M. Sypesteyn, J.W. Hoftijzer, S.C. Pont, F. Durand, and A. Bousseau. OpenSketch: a richly-annotated dataset of product design sketches. In *ACM Transactions on Graphics*, 2019. 3, 5

[13] B. Guillard, E. Remelli, and P. Fua. UCLID-Net: Single View Reconstruction in Object Space. In *Advances in Neural Information Processing Systems*, 2020. 1, 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[15] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a Sketching Interface for 3D Freeform Design. In *ACM SIGGRAPH 2006 Courses*, 2006. 2

[16] S. Ilić, M. Salzmann, and P. Fua. Implicit Meshes for Effective Silhouette Handling. *International Journal of Computer Vision*, 72(7), 2007. 3

[17] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-To-Image Translation with Conditional Adversarial Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017. 2, 4, 6

[18] A. Jin, Q. Fu, and Z. Deng. Contour-based 3D Modeling through Joint Embedding of Shapes and Contours. In *Symposium on Interactive 3D Graphics and Games*, 2020. 3

[19] Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. Sketching folds: Developable surfaces from non-planar silhouettes. *Acm Transactions on Graphics (TOG)*, 34(5):1–12, 2015. 2

[20] Olga Karpenko, John F Hughes, and Ramesh Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 2002. 2

[21] Youngihn Kho and Michael Garland. Sketching mesh deformations. In *ACM SIGGRAPH courses*, 2007. 2

[22] Y. G. Leclerc and M. Fischler. An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames. *International Journal of Computer Vision*, 9(2):113–136, 1992. 2

[23] T-.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755, 2014. 3

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

[25] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *ACM SIGGRAPH*, pages 163–169, 1987. 2

[26] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *International Conference on 3D Vision*, pages 67–77, 2017. 1, 3

[27] Jitendra Malik and Dror Maydan. Recovering three-dimensional shape from a single image of curved objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):555–566, 1989. 2

[28] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. Multi-class generative adversarial networks with the l2 loss function. *arXiv preprint arXiv:1611.04076*, 2016. 6

[29] T. Mcinerney and D. Terzopoulos. Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850, 1999. 2

[30] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2

[31] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *Conference on Computer Vision and Pattern Recognition*, pages 5425–5434, 2017. 2

[32] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH*, 2005. 2

[33] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. DeepSdf: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2

[34] O. Poursaeed, M. Fisher, N. Aigerman, and V.G. Kim. Coupling explicit and implicit surface representations for generative 3d modeling. In *European Conference on Computer Vision*, pages 667–683, 2020. 2, 3

[35] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *ACM SIGGRAPH Asia*, 2020. 4, 13

[36] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. Meshsdf: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, 2020. 1, 2, 3, 5, 8

[37] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241, 2015. 6

[38] M. Runz, K. Li, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, and R. Newcombe. Frodo: from Detections to 3D Objects. In *Conference on Computer Vision and Pattern Recognition*, June 2020. 2, 3

[39] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, 1990. 5

[40] M. Salzmann and P. Fua. Reconstructing Sharply Folding Surfaces: A Convex Formulation. In *Conference on Computer Vision and Pattern Recognition*, June 2009. 2

[41] J. A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999. 2

[42] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2003. 3

[43] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y. Jiang. Pixel2mesh: Generating 3D Mesh Models from Single RGB Images. In *European Conference on Computer Vision*, 2018. 1, 2, 3

[44] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction. In *Advances in Neural Information Processing Systems*, 2019. 1, 3, 5, 6, 8

[45] Yue Zhong, Yulia Gryaditskaya, Honggang Zhang, and Yi-Zhe Song. Deep Sketch-Based Modelling: Tips and Tricks. In *International Conference on 3D Vision*, 2020. 1, 6, 8

[46] Yue Zhong, Yonggang Qi, Yulia Gryaditskaya, Honggang Zhang, and Yi-Zhe Song. Towards practical sketch-based 3d shape generation: The role of professional sketches. In *IEEE Transactions on Circuits and Systems for Video Technology*, 2020. 1, 5