WIR3D: Visually-Informed and Geometry-Aware 3D Shape Abstraction

Richard Liu
University of Chicago
guanzhi@uchicago.edu

Daniel Fu
University of Chicago
danielfu@uchicago.edu

Noah Tan
University of Chicago
tntan@uchicago.edu

Itai Lang
University of Chicago
itai.lang83@gmail.com

Rana Hanocka
University of Chicago
ranahanocka@uchicago.edu

Figure 1. WIR3D produces 3D shape abstractions in the form of 3D strokes. The abstractions retain the overall shape structure and capture texture concepts (*e.g.* dragon scales and watermelon seeds) as well as key salient features (*e.g.* facial features).

Abstract

In this work we present WIR3D, a technique for abstracting 3D shapes through a sparse set of visually meaningful curves in 3D. We optimize the parameters of Bézier curves such that they faithfully represent both the geometry and salient visual features (e.g. texture) of the shape from arbitrary viewpoints. We leverage the intermediate activations of a pre-trained foundation model (CLIP) to guide our optimization process. We divide our optimization into two phases: one for capturing the coarse geometry of the shape, and the other for representing fine-grained features. Our second phase supervision is spatially guided by a novel localized keypoint loss. This spatial guidance enables user control over abstracted features. We ensure fidelity to the original surface through a neural SDF loss, which allows the curves to be used as intuitive deformation handles. We successfully apply our method for shape abstraction over a broad dataset of shapes with varying complexity, geometric structure, and texture, and demonstrate downstream applications for feature control and shape deformation.

1. Introduction

In this work, we explore whether it is possible to abstract a 3D shape into a sparse set of semantically-informed curves. A key challenge lies in finding a sparse set of curves that best represents the shape's *visual features*. We use this term deliberately, to encompass both the geometry and texture features which are salient to humans. This problem cannot be solved with surface analysis alone, which focuses on low-level geometric contours. Our task is much broader, in that we wish to capture high-level concepts, visually salient geometry, and textures (e.g. Fig. 1).

Existing works have dealt with the problem of computing occluding contours for non-photorealistic rendering [5, 29, 31]. Occluding contours relies entirely on surface analysis, which is subject to the aforementioned limitations. Furthermore, occluding contours is a 2D representation, while we specifically seek a view-consistent, 3D representation. Occluding contours is *view-inconsistent* by construction, which results in the commonly observed flickering artifact when rendering dense views of a 3D shape [21].

Our objective is to *abstract visual shape features into a sparse set of 3D strokes*. We optimize the parameters of a set of Bézier curves as our stroke representation, where the number of curves determines the level of abstraction.

We show a motivating example in Fig. 2, where we abstract a cylinder with a partial texture, and compare against a naive solution of back-projecting 2D occluding contour curves. The naive solution calculates a geometric contour at each view (inset), backprojects the result to 3D curves,

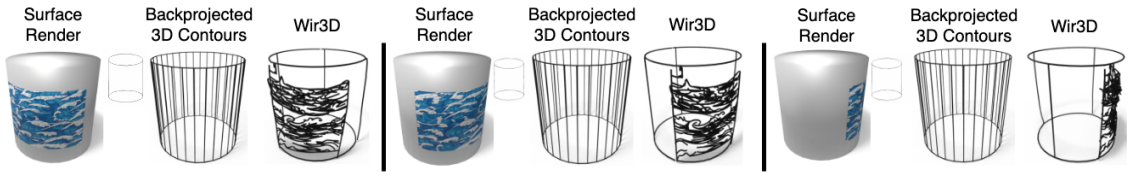

Figure 2. **Motivating example.** Abstracting even a simple shape like a cylinder is nontrivial. Rendering the contours of the cylinder (inset) results in a static image from every view, removing any sense of 3D volume. Backprojecting these contours into 3D results in a dense cluster of lines spanning the body of the cylinder, which is both non-sparse and unsatisfying aesthetically. In contrast, our method effectively abstracts the 3D geometry of the cylinder with a few strokes along with the texture.

and aggregates across all views. The naive solution only accounts for the cylinder geometry in a view-dependent manner, resulting in a dense body of strokes. Importantly, the naive solution cannot handle the texture at all.

Our solution represents the cylinder volume with sparse vertical curves. Different views of the shape are visually distinguished, yet the cylinder's overall geometry is consistent and clear from any given view. Our method also preserves the high-level pattern of the texture.

To achieve our sparse and visually salient abstraction, we leverage spatially-conditioned semantic supervision by utilizing the intermediate activations of a 2D pre-trained foundation model (CLIP [47]). Our choice of model is justified by prior work demonstrating CLIP's strong performance in tasks requiring high-level abstract understanding across different modalities [16, 18, 60]. We observe that both different layers and architectures enable different levels of control over geometric and texture elements of the shape and thus split our optimization into separate geometry and texture abstraction stages.

Our spatial conditioning comes from 3D keypoints, which determine the weight that specific visual features are given in our abstraction. The keypoints being in 3D ensures their multi-view consistency and also enables fine-grained user control over the resulting abstraction. We show in Fig. 9 an application where users can iteratively add detail to the abstraction through selecting keypoints corresponding to salient features.

We further encourage curve adherence to the input geometry using a neural SDF loss. The surface compliance enables an application where the curves can be used as intuitive deformation handles for the shape. We demonstrate this application in Fig. 8 and the supplemental, where our curves' effectiveness as control handles stems from their alignment to semantically meaningful regions of the surface and texture.

In summary, we present a novel technique for abstracting 3D shapes through a sparse set of visually-informed 3D curves. WIR3D can abstract a myriad of shapes from different domains with various visual concepts, geometric structures, and textures. The abstractions are sparse yet effective and maintain high fidelity across arbitrary views. Our novel localized weighting framework enables interactive user control over the features represented in the abstraction, and our adherence to the input geometry allows for the curves' use as intuitive deformation handles. We encourage the reader to examine the supplemental material, which contains 360-degree videos of our results and an interactive demo of the deformation application using our optimized curves. We plan to release the code for the method and proposed applications in the near future.

2. Related Work

Shape decomposition. Shape decomposition is a long-standing problem in 3D geometry analysis, where the goal is to represent the shape by a set of elements, such as primitives [34, 53, 57, 65], convex parts [12, 24, 39, 44], or Gaussians [19, 20]. Tulsiani *et al*. [57] assemble 3D objects from cuboids and obtained simple shape abstractions with consistent structure. In Cvxnet [12], the authors reconstruct a 3D shape by a collection of convex polytopes. Paschalidou *et al*. [43] extend this approach by learning the decomposition and the parts for a set of domain-specific meshes.

This line of work aims at *reconstruction*, whereas our goal is *abstraction*. Moreover, our method does not depend on a dataset and is not limited to a specific domain. It is robust to shapes of arbitrary quality and complexity.

Non-photorealistic rendering. A classic problem in graphics is identifying visual contours from 3D geometry to create non-photorealistic renderings. A popular version of this problem is occluding contours [1, 2, 10, 25, 36, 48], in which contours representing visible shape regions are delineated from occluded ones. In the classic setting, visible contours are exactly defined to mean the visible surface points tangent to the view vector. Recent efforts have aimed at improving view-consistency and alignment with professional artists [5], and some apply neural techniques [30, 31]. While such works can successfully depict visual features based on the shape geometry, they lack 3D consistency, and frequently suffer from flickering artifacts [21].

A recent paper [67] and its follow-up [26] introduce implicit edge fields for 3D curve reconstruction from posed

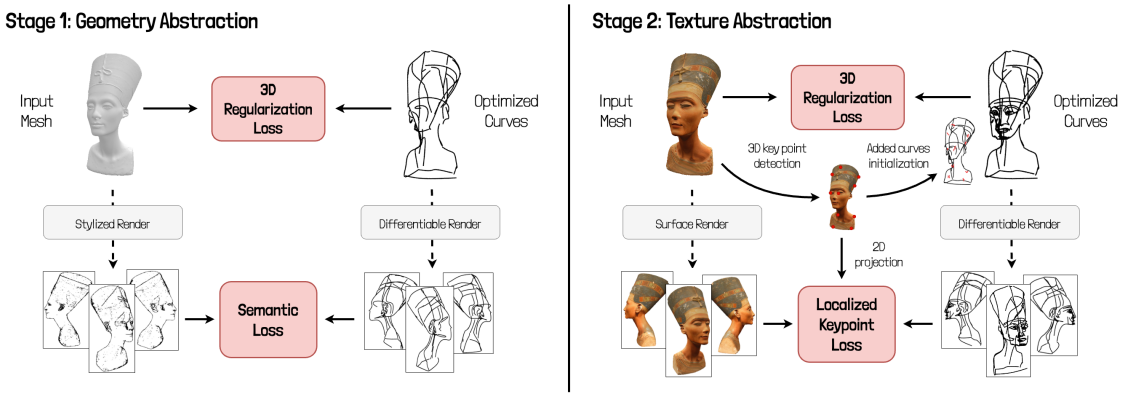

Figure 3. **Method overview.** In the first stage, our WIR3D learns to abstract the underlying geometry of the shape. In the second stage, we freeze the curves from the first stage and add new curves that are optimized to represent the shape's texture.

images. These works emphasize geometric boundaries, which is distinct from our focus on *abstraction*. A high-level abstraction may reduce a shape feature to a single stroke (e.g. Fig. 6 chair), which is not achievable by these methods. Furthermore, these methods cannot account for texture features which do not induce a prominent edge map.

Curve-based abstraction. Our work is in the domain of sketch abstraction, which aims to depict a scene through 2D or 3D curve primitives. [7, 14, 22, 37, 59]. Most prior work is based on 2D sketch abstractions [6, 13, 15, 23, 27, 42, 45, 51, 54, 59, 60], though several works have aimed to represent 2D images through a 3D structures by projection along orthogonal axes [22, 38, 46]. Our task is different in that we aim to represent a single shape from all possible viewing directions, not simply three orthogonal views.

Another line of work aims to reconstruct fabricable 3D wire structures from different modalities, including images [8, 32, 33], video [62], and 3D data [4, 55, 66]. These works are constrained by the fabrication objective, and thus do not abstract fine-grained visual details.

A final strain of literature deals with analysis of 2D sketches of 3D structures to identify part information and aid in user sketch generation [17, 41, 49].

A recent work proposed optimizing strokes in 3D to match a text description or a guidance image [69]. This work is generative in nature (*e.g.* text-to-3D sketch) and does not involve spatial nor 3D inputs as in our work. 3Doodle [7] presents a technique for optimizing view-dependent and view-independent curves to represent a set of multi-view images. Our work is orthogonal in spirit – we aim to use a set of view-independent curves for 3D *abstraction* of an *input shape*, and we build our framework leveraging the geometric and semantic shape information. Notably, our use of spatially-driven guidance and SDF loss enables our detail control and deformation applications.

We use a variant of 3Doodle with only view-independent curves as our main baseline and show results in Fig. 6.

3. Method

WIR3D optimizes a set of 3D cubic Bézier curves to abstract a target (potentially textured) shape from all viewing angles. The method takes as input a 3D model and optional user-selected keypoints on the surface. When no keypoints are provided, we automatically detect keypoints using latent backprojection and clustering (Sec. 3.3).

3.1. Curve Representation

Our 3D strokes are modeled as a set of cubic Bézier curves $\{B_i\}_{i=1}^{n}$, with control points $B_i = (p_i^0, p_i^1, p_i^2, p_i^3)$, $p_i^j \in \mathbb{R}^3$. Points on the curve are sampled through polynomial interpolation of the four control points

$$\begin{aligned} B(t) =&(1-t)^3 p^0 + (1-t)^2 t p^1 \\ &+ (1-t) t^2 p^2 + t^3 p^3 \end{aligned} \tag{1}$$

where $0 \leq t \leq 1$.

We make the same assumption as 3Doodle [7] that the camera is sufficiently far from the shape such that orthographic and perspective projection are nearly identical. Theorem 1 from 3Doodle thus applies to our method, which establishes equivalence between the normal 3D Bézier curves we optimize and the space of 2D rational Bézier curves, generated from projecting the 3D curve control points into 2D. Our pipeline renders the 3D Bézier curves by first perspective-projecting the 3D control points, then rasterizing the 2D cubic Bézier defined from the projected control points (interpolated as in Eq. (1)). This process is encapsulated by the "Differentiable Render" step in Fig. 3. The differentiable rasterizer we use is DiffVG [28].

The resulting image from this projection and rasterization process is $I_{\text{curve}} = \mathcal{R}(\{B_i\})$. We refer to target shape renders as I_{target}.

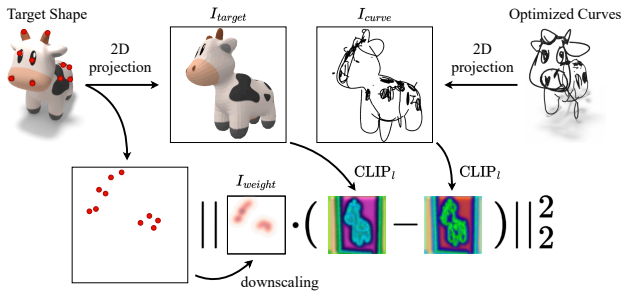

Figure 4. **Localized keypoint loss.** Our localized keypoints weight the loss between the intermediate feature maps of the encoded curve render I_{curve} and the target shape render I_{target}. This weight is obtained through projecting 3D keypoints (red), followed by a Gaussian filter to obtain the weight map I_{weight}. This loss focuses the optimization on visual features local to the keypoint.

3.2. Losses

We leverage the priors of 2D pretrained image encoders to define our semantic losses. Specifically, we design specialized perceptual losses using CLIP [47] and LPIPS [68] to encourage our rendered 3D strokes to visually match the corresponding renders of the target shape.

The basic structure of our semantic loss is adapted from CLIPasso [59], which compares both the intermediate spatial activations and final global activations between the rendered strokes and the target shape:

$$\begin{aligned}\mathcal{L}_{\text{semantic}} =& \lambda_{\text{fc}}\text{dist}(\text{CLIP}(I_{\text{curve}}), \text{CLIP}(I_{\text{target}})) \\ &+ \sum_{l=3,4} ||\text{CLIP}_l(I_{\text{curve}}) - \text{CLIP}_l(I_{\text{target}})||_2^2\end{aligned} \tag{2}$$

where $\text{dist}(\cdot,\cdot)$ measures cosine similarity, CLIP is the global CLIP encoding, and CLIP_l is the layer l activation.

We find this semantic loss, however, to be lacking when it comes to representing fine details, including textures. Thus, we introduce a *spatial weighting* framework, which directs the optimization towards specific visual features.

Localized Keypoint Loss. Our spatial weighting is based on previous work that establishes the spatial correlation between CLIP's intermediate activations and the input image. [50]. Features of interest can be emphasized in these intermediate layers by identifying their location in the render and tracing the correspondence to the downsampled feature maps, hence *localizing* the features in the feature maps.

We assume as optional input user-defined 3D keypoints. If no input keypoints are given then our method automatically detects keypoints, as described in Sec. 3.3 and the supplemental. These keypoints should indicate salient visual features on the input shape or texture, and we leverage that information to guide abstraction of the specific features.

For every sampled view during optimization, we project the keypoints to the same views and identify their positions in the corresponding renders. Once we have the keypoint locations in the rendered image, we construct a weight map at the same resolution as the image based on a Gaussian dropoff from the keypoint center. Specifically, we construct the weight image I_{weight} such that

$$I_{\text{weight}}(x,y) = 1 + \sum_p e^{\frac{-||(x,y)-p||^2}{2\sigma^2}} \tag{3}$$

where $x, y \in [0,1]$ index the normalized image pixels, and $p \in [0,1]$ is the keypoint projected to normalized coordinates. We add 1 to the weights so that regions far from the keypoints still contribute to the loss. We use these constructed weight maps to weight our semantic loss Eq. (2) for each render, such that each L2 term in the intermediate layer losses $||\text{CLIP}_l(I_{\text{curve}}) - \text{CLIP}_l(I_{\text{target}})||_2^2$ becomes $||I_{\text{weight}} \cdot (\text{CLIP}_l(I_{\text{curve}}) - \text{CLIP}_l(I_{\text{target}}))||_2^2$. The weight map is downsampled to match the resolution of each intermediate feature map. We visualize this process in Fig. 4.

We maintain a z-buffer for the shape renders, such that if a keypoint's projected depth is higher than the shape depth, then the keypoint is occluded by the surface from that view and does not contribute to the weight image. This prevents keypoints from being attributed to incorrect surface regions.

To provide additional structural constraints, we include an LPIPS loss term [68]. LPIPS is a popular perceptual loss function which is known to be sensitive to geometric layouts [7]. The final localized keypoint loss becomes:

$$\begin{aligned}\mathcal{L}_{\text{local}} =& \lambda_{\text{fc}}\bar{I}_{\text{weight}}\text{dist}(\text{CLIP}(I_{\text{curve}}), \text{CLIP}(I_{\text{target}})) \\ &+ \sum_{l=3,4} ||I_{\text{weight}} \cdot (\text{CLIP}_l(I_{\text{curve}}) - \text{CLIP}_l(I_{\text{target}}))||_2^2 \\ &+ \lambda_{\text{lpips}}\text{LPIPS}(I_{\text{curve}}, I_{\text{target}})\end{aligned} \tag{4}$$

where $\bar{I}_{\text{weight}}$ indicates the mean-pooled weights.

A natural question is how sensitive this loss is to poor keypoint selection. We show an example optimization using our localization loss with randomly distributed keypoints in Fig. 18 the supplemental, and show the result is not much different than optimizing without any keypoints ($\mathcal{L}_{\text{semantic}}$). We also ablate on $\mathcal{L}_{\text{local}}$ when keypoints identify salient features in Fig. 17, and show the localization weighting is critical for abstracting fine-grained features. This demonstrates that $\mathcal{L}_{\text{local}}$ is a strict improvement over $\mathcal{L}_{\text{semantic}}$ in cases where the keypoints are meaningful, and otherwise produces robust results on par with $\mathcal{L}_{\text{semantic}}$.

On top of the localized keypoint loss, we include SDF and view regularization losses to ensure the stroke abstraction is represented in all possible views.

SDF Regularization. To encourage adherence of the curves to the target shape geometry, we use a loss based

on the shape's Signed Distance Field (SDF). We fit an MLP Φ on the shape's SDF (Unsigned Distance Field in the case of shapes with boundaries) to obtain a neural SDF. During stroke optimization, we densely sample each 3D curve and query their SDF values using the neural SDF, penalizing values outside of the zero level set:

$$\mathcal{L}_{\text{SDF}} = \frac{1}{n \cdot k} \sum_{i=1}^{n} \sum_{k=1}^{s} |\phi(B_i(t_k)| \tag{5}$$

where $t_k \in [0, 1]$ are random samples along the Bézier curve for s total samples. This loss helps to anchor abstracted features to the surface implied by the curve set.

View Regularization. We further regularize the abstraction by enforcing that all curves are visible from all sampled viewing angles. This forces all curves to participate in the shape abstraction from every angle, which is necessary for a proper 3D abstraction. With $\mathcal{P}$ indicating the perspective projection of the Bézier curve control keypoints, we have:

$$\mathcal{L}_{\text{ndc}} = \sum_{i=1}^{n} \sum_{j=1}^{2} \text{ReLU}(\mathcal{P}(B_i)_j - 1) + \text{ReLU}(-\mathcal{P}(B_i)_j), \tag{6}$$

where ReLU is a Rectified Linear Unit.

3.3. Two-Stage Optimization

In our experiments we find that different CLIP architectures and layer combinations are sensitive to geometry or semantic shape features. To exploit this, we construct a two-stage training pipeline, in which the first stage is optimizes for the shape *geometry* and the second stage abstracts the shape *texture*, where each stage leverages different CLIP architectures and semantic losses.

Keypoint Initialization. When keypoints are not included in the input, we automatically identify keypoints of interest on the shape's surface using the 2D to 3D feature back-projection method introduced in Backto3D [64] with KMeans clustering [35]. See supplemental for more details.

Geometry Abstraction. During stage I optimization, the Bézier curves are initialized using furthest point sampling, with the control points drawn from small Gaussians around each sampled point. These curves are optimized towards the shape geometry with a combination of our original (non-localized) semantic (2), SDF (5), and NDC (6) losses, supervised with Freestyle renders (see supplemental) $I_{\text{target}}^{\text{free}}$ of the target shape. The stage I loss is:

$$\mathcal{L}_{\text{I}} = \mathcal{L}_{\text{semantic}}(I_{\text{curve}}, I_{\text{target}}^{\text{free}}) + 0.1 \cdot \mathcal{L}_{\text{SDF}} + \mathcal{L}_{\text{ndc}}(\{B_i\}).$$

Texture Abstraction. In our second stage, we freeze the set of geometry curves optimized in the first stage and initialize a new set of curves in the same way as the first stage, using furthest distance sampling. These curves are then optimized to represent the semantic texture of the shape using our localized keypoint loss (4), SDF (5), and NDC (6) losses, supervised with surface (potentially textured) renders of the shape $I_{\text{target}}^{\text{surface}}$. The stage II loss is:

$$\mathcal{L}_{\text{II}} = \mathcal{L}_{\text{local}}(I_{\text{curve}}, I_{\text{target}}^{\text{surface}}) + \mathcal{L}_{\text{SDF}} + \mathcal{L}_{\text{ndc}}(\{B_i\}).$$

We use CLIP architectures RN101 and RN50x16 for $\mathcal{L}_{\text{semantic}}$ and $\mathcal{L}_{\text{local}}$, respectively. We've found empirically that RN101 tends to be more sensitive to geometric structures, whereas RN50x16 is more sensitive to higher-level visual concepts.

4. Experiments

We evaluate WIR3D across a wide variety of shapes and demonstrate multi-view fidelity and abstraction control in Sec. 4.1. We compare WIR3D quantitatively and qualitatively to relevant baselines in Sec. 4.2. Finally, Sec. 4.4 showcases the interactive feature control and shape deformation applications enabled by our method.

The shapes in our experiments are aggregated from COSEG [58], the Meta Digital Twin Catalog [40], and Keenan Crane's 3D model repository [9]. WIR3D is robust to meshes with varying topology, complexity and quality.

For all experiments, we randomly sample views spanning 0 to 30 degrees elevation and 0 to 360 degrees azimuth.

4.1. Qualitative Results

Shape Abstractions. Fig. 5 shows 3D stroke abstractions generated by WIR3D for textured shapes.WIR3D also successfully abstracts textures and captures structured patterns, such as watermelon seeds (Row 1) and spots (Rows 5/6 Spot/Bob), as well as facial features (Rows 5/7 Spot/Nefertiti). We also show results on untextured shapes in Fig. 13 in the supplemental. Our method can represent complex geometries, such as the spiral column band (Row 6), the parallel rows of spines on the stegosaurus (Row 5).

Abstraction Control. Our method automatically adapts the level of abstraction based on the number of strokes being optimized. Fig. 7 shows that adding more strokes adds progressvely more detail to the abstraction, and our method produces high quality abstractions across all levels.

Multi-View Fidelity. Because our curves are defined in 3D, our abstraction is view-consistent by construction. However, this does not guarantee the curves plausibly represent the shape from arbitrary views. Fig. 14 in the supplemental shows that our abstraction faithfully represents the shape for densely sampled views in a 360 range.

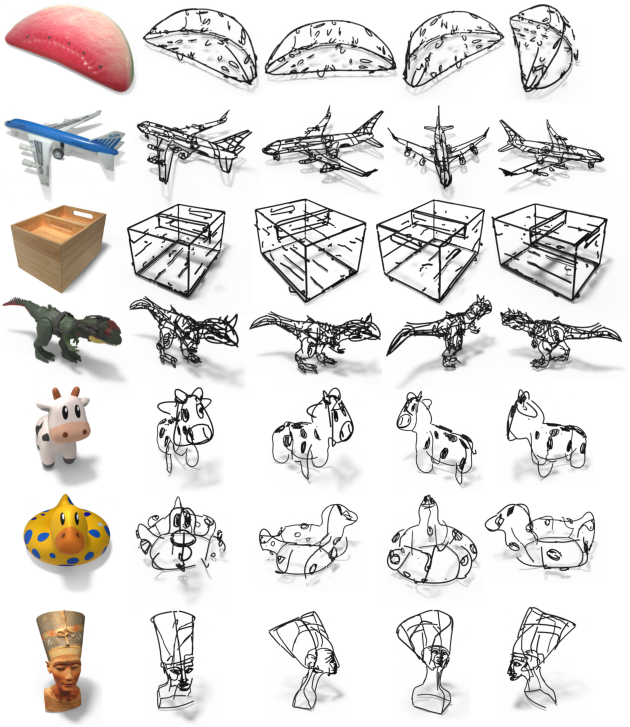

Figure 5. **Qualitative results for textured objects.** We show WIR3D's the result on a collection of textured meshes.

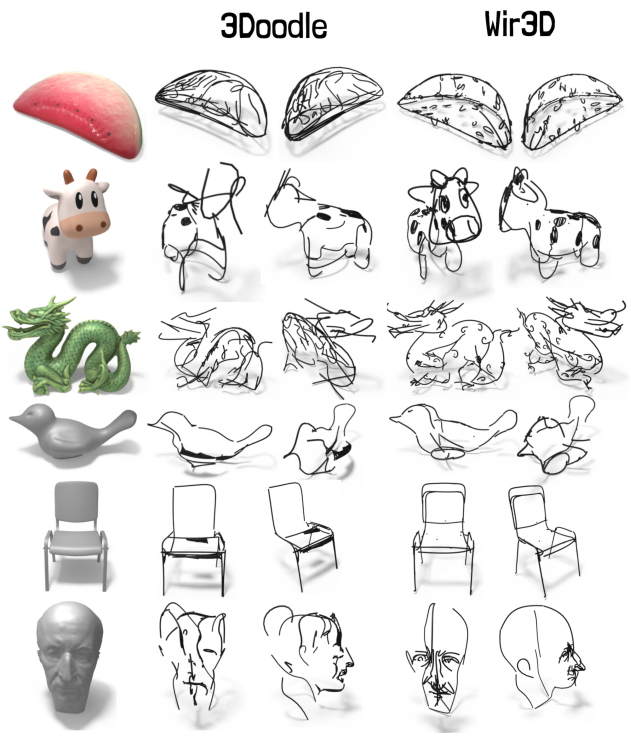

Figure 6. **Qualitative comparison.** We compare a subset of our textured and untextured results against 3Doodle. 3Doodle produces reasonable abstractions that capture the overall structure but lack precision when it comes to the feature details and specific geometric structures.

4.2. Baseline Comparisons

The baselines against which we compare are NEF [67] and 3Doodle [7]. For both methods, we use the publicly available repositories published by the respective authors. We standardize the sampled views during optimization, and use the default hyperparameters otherwise. 3Doodle originally initializes its curves using SFM, but given the method's sensitivity to poor initialization, we standardize initialization to the same furthest point samples as our method, and find that they perform better than SFM.

NEF often struggles to fit reasonable point clouds, which results in meaningless curves. We show qualitative comparison to NEF in the supplementary, where we train NEF on a set of viewpoints optimized for the method.

Qualitative Comparison. We show qualitative comparisons with the 3Doodle baseline for both untextured and textured shapes in Fig. 6. Our method consistently captures the global geometry, whereas 3Doodle struggles in a low signal context such as untextured shapes (missing back left chair leg, flattened bird geometry). Furthermore, 3Doodle is consistently insensitive to facial structures, whereas our localized weighting ensures that we abstract them (rows 2, 6). Similarly, we are able to represent fine-grained texture structures as high-level visual motifs, such as the dragon scales (row 3).

Quantitative Comparison. We assess the quality of the stroke abstractions using three perceptual metrics and one geometric metric, reported in Tab. 1. The first two, LPIPS [68] and CLIP$^{\text{img}}$ [61], are the same metrics reported by 3Doodle. Our method outperforms 3Doodle on both metrics, though numerically our CLIP$^{\text{img}}$ is not much higher. We note that CLIP has been shown to be insensitive to large visual differences, as systematized in [56, 63]. We further illustrate this point in Fig. 12 (supplemental), where we show that a randomly oriented cow outline obtains a similar CLIP$^{\text{img}}$ score as the edge map [3] of the *ground-truth render*. Thus, the numerical difference in CLIP$^{\text{img}}$ is an unreliable indicator of the difference in visual quality.

This motivates our user perceptual study ($N = 96$), which takes all the results from our dataset and compares our Wir3D optimized strokes against the 3Doodle optimized strokes. Specifically, we display rotating gifs of both results side by side, along with a rotating gif of the target shape, and ask users to rank the 3D strokes based on how well they represent the target shape. We collected responses from 96 users and compute the frequency our method is ranked higher than 3Doodle. Our curves are ranked as bet-

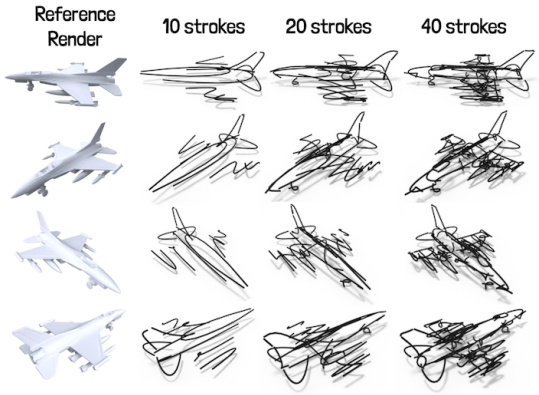

Figure 7. **Abstraction control.** The level of abstraction is implicitly controlled by the number of curves. As the curve count grows, the abstraction captures more fine details.

ter shape abstractions 88% of the time. Screenshots from the study are shown in the supplemental.

Our final metric, "Coverage", is a geometric metric that quantifies how well the stroke abstractions cover the original surface. We aim to abstract an input 3D model, so fidelity to the input geometry is key to abstraction quality. We measure coverage by sampling 100k points over the input surface and computing the 1-direction Chamfer distance from the surface points to the optimized strokes. This metric evaluates whether each point on the surface has a curve reasonably close to it. Adequate coverage of a shape's geometry is important for downstream applications which make use of the surface correspondence, such as the deformation application shown in Fig. 8. Our method significantly outperforms the baselines in terms of coverage, with a >2x reduction in coverage distance relative to 3Doodle, and >4x reduction relative to NEF.

Method	LPIPS (↓)	CLIP^{img} (↑)	User Rank (↑)	Coverage (↓)
NEF	0.313	0.86	-	0.056
3Doodle	0.246	0.900	0.12	0.020
WIR3D (Ours)	**0.227**	**0.909**	**0.88**	**0.008**

Table 1. We compare between WIR3D, 3Doodle [7], and NEF [67] through 3 perceptual metrics computed over novel views of the curves and the target shape. "LPIPS" measures the average LPIPS similarity score using AlexNet. "CLIP^{img}" measures the average cosine similarity scaled between (0,1) using the ViT/B-32 model. "User Rank" measures the percentage of user responses that prefer one method over the other ($N = 96$). We also compute a geometric metric "Coverage", which measures the 1-direction Chamfer distance between the curves and the target 3D surface.

4.3. Ablations

We perform a thorough ablation study demonstrating the importance of our design choices in WIR3D. Figures for each ablation are shown in the supplemental. We report the ablation metrics in Tab. 2. Removing the CLIP intermediate layers from our supervision results in the highest reduction in quality, followed by removing the stage 1 training. Removing the localized keypoint loss also results in some reduction in the metrics, but as discussed in Sec. 4.2, perceptual metrics may not be sensitive to the fine visual detail this loss is designed to capture. Removing the SDF loss results in worse geometric coverage, as expected.

Stage 1. Stage 1 training ensures the full geometry is represented from every view. Removing this stage results in the optimization biasing towards certain views and creating an overall "flattened" effect of the geometry (Fig. 15).

No intermediate CLIP layers. As established in [59], the intermediate CLIP layers are essential for capturing the geometric structure of the target. Optimizing on only the fully-connected CLIP output results in abstractions that have some semantic correspondence with the target but the specific geometric features are noisy (Fig. 16).

Localized keypoint loss. As discussed in Sec. 3.2, our method without keypoints produces reasonable abstraction but lacks fine-grained detail (Fig. 17). We also show in Fig. 18 that initialization with random keypoints is no worse than running our method without keypoints. Hence, we are robust to poor keypoint choice.

SDF loss. Our SDF loss prevents semantic features from pulling off the surface implied by the geometry curves. It also prevents occasional small floaters (Fig. 19).

Method	LPIPS (↓)	CLIP^{img} (↑)	Coverage (↓)
WIR3D	**0.227**	**0.909**	**0.008**
No SDF	0.229	0.904	0.012
No Local	0.233	0.905	0.009
w/o Stage 1	0.248	0.900	0.016
No CLIP Layers	0.294	0.891	0.012

Table 2. **Ablation Quantitative Metrics.** "WIR3D" is our full method. "w/o Stage 1" is the Stage 1 ablation. "No CLIP layers" is the ablation on CLIP intermediate layer activations. "No SDF" is the ablation on the SDF loss. "No Local" is the ablation on the localized keypoint loss.

4.4. Applications

We demonstrate two applications enabled by our 3D curve representation. The first is user-interactive feature control. Our localized weighting framework allows for user control over which features are represented in the abstraction. After

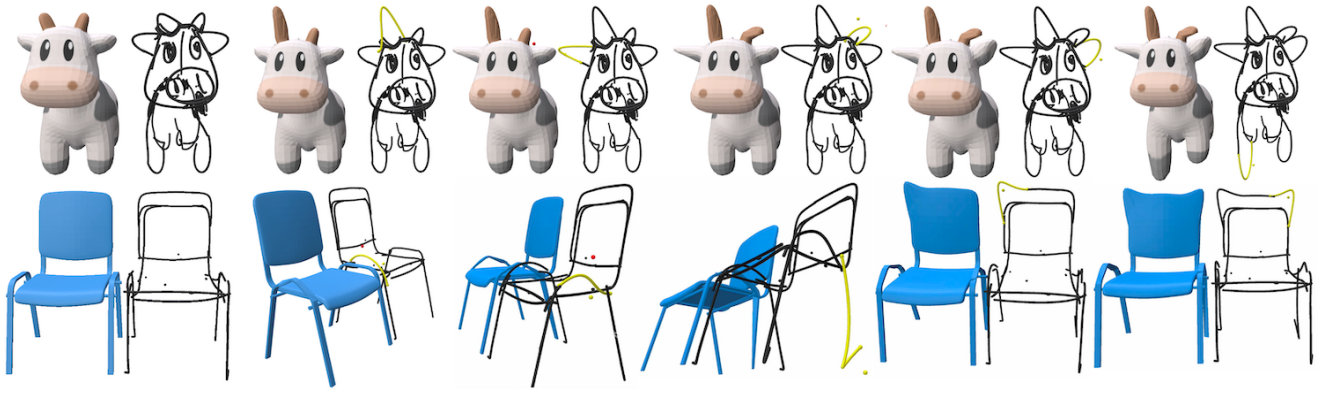

Figure 8. **Deformation application.** The curve abstractions make for intuitive control handles for shape deformation. The curves wrap salient features such that deformations are transferred smoothly through a simple L2 distance skinning procedure.

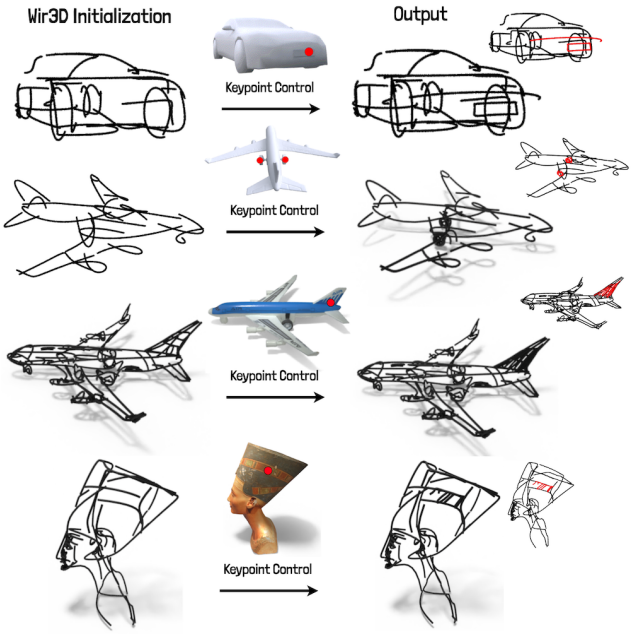

Figure 9. **Keypoint control.** Our spatial weighting framework enables user-interactive detail refinement.

optimization, the user can further refine the curves by selecting additional keypoints, and new curves can be quickly optimized to add detail to the feature of interest. We demonstrate this refinement procedure in Fig. 9, where keypoints can be used to make specific structures more explicit (e.g. wheels on the plane) or to add texture detail (e.g. nefertiti headband). The refinement is rapid, and is completed in a few hundred iterations, or around a minute.

The curves' adherence to the input geometry makes them intuitive deformation handles. We show in Fig. 8 examples of interactive deformation using the curves, where Euclidean distance-based skinning weights are used to map deformations from the curves to the shape surface. The curves naturally wrap salient features so that the deformations are smoothly transferred, and relevant parts are easily manipulated. More detail and videos are in the supplemental.

5. Conclusion

WIR3D is a method for 3D model abstraction into sparse strokes, parameterized as cubic Bézier curves in 3D. We introduce a novel localized keypoint loss, which allows the abstractions to represent fine-grained geometry and texture details. This localized weighting framework enables user control over the local abstraction detail through interactive keypoint selection and detail refinement. Our method is robust across models of arbitrary topology and quality. We encourage the 3D curves to maintain close correspondence to the original surface, which can enable intuitive shape editing applications such as curve-based deformation.

Limitations. The quality of WIR3D is primarily limited by the quality of the detected/input keypoints. In cases where the keypoints are located in non-semantic regions, our abstraction will perform on par with the result without localized weighting. Another limitation is the preprocessing required for our method. Fitting a neural SDF, running our automatic keypoint algorithm, and generating the freestyle renders may take a lot of time depending on the complexity of the input model. For instance, the total preprocessing time for our Nefertiti model (100,000 faces) is around 2 hours. Future work could look into making this preprocessing more efficient or removing it entirely, and producing better algorithms for semantic keypoint detection.

References

[1] Edmond Boyer and Marie-Odile Berger. 3D Surface Reconstruction Using Occluding Contours. *International Journal of Computer Vision*, 22(3):219–233, 1997.

[2] Pierre Bénard and Aaron Hertzmann. Line Drawings from

3D Models: A Tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 11(1-2):1–159, 2019.

[3] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[4] Li Cao, Yike Xu, Jianwei Guo, and Xiaoping Liu. WireframeNet: A novel method for wireframe generation from point cloud. *Computers & Graphics*, 115:226–235, 2023.

[5] Ryan Capouellez, Jiacheng Dai, Aaron Hertzmann, and Denis Zorin. Algebraic Smooth Occluding Contours. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023.

[6] Hong Chen, Ying-Qing Xu, Heung-Yeung Shum, Song-Chun Zhu, and Nan-Ning Zheng. Example-based facial sketch generation with non-parametric sampling. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, pages 433–438 vol.2, 2001.

[7] Changwoon Choi, Jaeah Lee, Jaesik Park, and Young Min Kim. 3Doodle: Compact Abstraction of Objects with 3D Strokes. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024.

[8] Hyelim Choi, Minji Lee, Jiseock Kang, and Dongjun Lee. Online 3D Edge Reconstruction of Wiry Structures From Monocular Image Sequences. *IEEE Robotics and Automation Letters*, 8(11):7479–7486, 2023. Conference Name: IEEE Robotics and Automation Letters.

[9] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[10] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[11] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive Contours for Conveying Shape. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 401–408. ACM New York, NY, USA, 2023.

[12] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable Convex Decomposition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 31–44, 2020.

[13] Kevin Frans, Lisa Soros, and Olaf Witkowski. CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders. *Advances in Neural Information Processing Systems*, 35:5207–5218, 2022.

[14] Ran Gal, Olga Sorkine, Niloy Mitra, and Daniel Cohen-Or. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 28(3):33:1–33:10, 2009.

[15] Rinon Gal, Yael Vinker, Yuval Alaluf, Amit Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. Breathing Life Into Sketches Using Text-to-Video Priors. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4325–4336, Seattle, WA, USA, 2024. IEEE.

[16] Huy Ha and Shuran Song. Semantic Abstraction: Open-World 3D Scene Understanding from 2D Vision-Language Models, 2022. arXiv:2207.11514 [cs].

[17] James W. Hennessey, Han Liu, Holger Winnemöller, Mira Dontcheva, and Niloy J. Mitra. How2sketch: Generating easy-to-follow tutorials for sketching 3d objects. *Symposium on Interactive 3D Graphics and Games*, 2017.

[18] Pablo Hernandez-Camara and Jorge Vila-Tomas. MEASURING HUMAN-CLIP ALIGNMENT AT DIFFERENT ABSTRACTION LEVELS. 2024.

[19] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. PointGMM: a Neural GMM Network for Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12054–12063, 2020.

[20] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–20, 2022.

[21] Aaron Hertzmann. New insights in smooth occluding contours for nonphotorealistic rendering. *IEEE Computer Graphics and Applications*, 44(1):76–85, 2024.

[22] Kai-Wen Hsiao, Jia-Bin Huang, and Hung-Kuo Chu. Multi-view Wire Art. *ACM Transactions on Graphics (TOG)*, 37 (6):1–11, 2018.

[23] Felix Hähnlein, Changjian Li, Niloy J. Mitra, and Adrien Bousseau. CAD2Sketch: Generating Concept Sketches from CAD Sequences. *ACM Transactions on Graphics*, 41(6):1–18, 2022.

[24] R Kenny Jones, Aalia Habib, and Daniel Ritchie. SHRED: 3D Shape Region Decomposition with Learned Local Operations. *ACM Transactions on Graphics (TOG)*, 41(6):1–11, 2022.

[25] Jan J Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984.

[26] Lei Li, Songyou Peng, Zehao Yu, Shaohui Liu, Remi Pautrat, Xiaochuan Yin, and Marc Pollefeys. 3D Neural Edge Reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21219–21229, Seattle, WA, USA, 2024. IEEE.

[27] Mengtian Li, Zhe Lin, Radomír Mˇ ech, , Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. *WACV*, 2019.

[28] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[29] Chenxi Liu, Pierre Bénard, Aaron Hertzmann, and Shayan Hoshyari. ConTesse: Accurate Occluding Contours for Subdivision Surfaces. *ACM Transactions on Graphics*, 42(1): 1–16, 2023.

[30] Difan Liu, Mohamed Nabail, Aaron Hertzmann, and Evangelos Kalogerakis. Neural Contours: Learning to Draw Lines from 3D Shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5428–5436, 2020.

[31] Difan Liu, Matthew Fisher, Aaron Hertzmann, and Evangelos Kalogerakis. Neural Strokes: Stylized Line Drawing of 3d Shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 14204–14213, 2021.

[32] Lingjie Liu, Duygu Ceylan, Cheng Lin, Wenping Wang, and Niloy J. Mitra. Image-based reconstruction of wire art. *ACM SIGGRAPH 2017*, 2017.

[33] Lingjie Liu, Nenglun Chen, Duygu Ceylan, Christian Theobalt, Wenping Wang, and Niloy J. Mitra. Curvefusion: Reconstructing thin structures from rgbd sequences. *ACM Trans. Graph.*, 37(6):218:1–218:12, 2018.

[34] Weixiao Liu, Yuwei Wu, Sipu Ruan, and Gregory S Chirikjian. Marching-Primitives: Shape Abstraction from Signed Distance Function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8771–8780, 2023.

[35] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[36] David Marr. Analysis of Occluding Contour. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 197(1129):441–475, 1977.

[37] Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J Mitra. Abstraction of man-made shapes. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–10. ACM New York, NY, USA, 2009.

[38] Niloy J Mitra and Mark Pauly. Shadow Art. *ACM Transactions on Graphics*, 28(5):156–1, 2009.

[39] Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Transactions on Graphics (TOG)*, 37(5):1–15, 2018.

[40] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception, 2023.

[41] Karran Pandey, Fanny Chevalier, and Karan Singh. Juxtaform: interactive visual summarization for exploratory shape design. *ACM Trans. Graph.*, 42(4):52:1–52:14, 2023.

[42] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. SketchGen: Generating Constrained CAD Sketches. In *Advances in Neural Information Processing Systems*, pages 5077–5088. Curran Associates, Inc., 2021.

[43] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3204–3215, 2021.

[44] Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A Yeh, and Rana Hanocka. GeoCode: Interpretable Shape Programs. *arXiv preprint arXiv:2212.11715*, 2022.

[45] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. SketchLattice: Latticed Representation for Sketch Manipulation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 933–941, Montreal, QC, Canada, 2021. IEEE.

[46] Zhiyu Qu, Lan Yang, Honggang Zhang, Tao Xiang, Kaiyue Pang, and Yi-Zhe Song. Wired Perspectives: Multi-View Wire Art Embraces Generative AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6149–6158, 2024.

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[48] W Brent Seales and Charles R Dyer. Viewpoint from Occluding Contour. *CVGIP: Image Understanding*, 55(2):198–211, 1992.

[49] Tianjia Shao, Wilmot Li, Kun Zhou, Weiwei Xu, Baining Guo, and Niloy J. Mitra. Interpreting concept sketches. *ACM Transactions on Graphics*, 32(4), 2013.

[50] Gil Shomron and Uri Weiser. Spatial correlation and value prediction in convolutional neural networks. *IEEE Comput. Archit. Lett.*, 18(1):10–13, 2019.

[51] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning to Sketch with Shortcut Cycle Consistency. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 801–810, Salt Lake City, UT, 2018. IEEE.

[52] Anjana Susarla, Ram Gopal, Jason Bennett Thatcher, and Suprateek Sarker. The Janus Effect of Generative AI: Charting the Path for Responsible Conduct of Scholarly Activities in Information Systems. *Information Systems Research*, 34 (2):399–408, 2023. Publisher: INFORMS.

[53] Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. Sphere-Meshes: shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics*, 32 (6):1–12, 2013.

[54] Yingtao Tian and David Ha. Modern Evolution Strategies for Creativity: Fitting Concrete Images and Abstract Concepts. In *Artificial Intelligence in Music, Sound, Art and Design: 11th International Conference, EvoMUSART 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*, pages 275–291, Berlin, Heidelberg, 2022. Springer-Verlag.

[55] Kenji Tojo, Ariel Shamir, Bernd Bickel, and Nobuyuki Umetani. Fabricable 3D Wire Art. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, pages 1–11, Denver CO USA, 2024. ACM.

[56] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs, 2024. arXiv:2401.06209 [cs].

[57] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning Shape Abstractions by Assembling Volumetric Primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017.

[58] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, and Ghassan Hamarneh. Prior knowledge for part correspondence. *Computer Graphics Forum*, 30(2):553–562, 2011.

[59] Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. CLIPasso: Semantically-Aware Object Sketching. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.

[60] Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. CLIPascene: Scene Sketching with Different Types and Levels of Abstraction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4123–4133, Paris, France, 2023. IEEE.

[61] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *AAAI*, 2023.

[62] Peng Wang, Lingjie Liu, Nenglun Chen, Hung-Kuo Chu, Christian Theobalt, and Wenping Wang. Vid2Curve: simultaneous camera motion estimation and thin structure reconstruction from an RGB video. *ACM Trans. Graph.*, 39(4): 132:132:1–132:132:12, 2020.

[63] Wenxuan Wang, Quan Sun, Fan Zhang, Yepeng Tang, Jing Liu, and Xinlong Wang. Diffusion Feedback Helps CLIP See Better, 2024. arXiv:2407.20171 [cs].

[64] Thomas Wimmer, Peter Wonka, and Maks Ovsjanikov. Back to 3d: Few-shot 3d keypoint detection with back-projected 2d features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[65] Yuwei Wu, Weixiao Liu, Sipu Ruan, and Gregory S Chirikjian. Primitive-based Shape Abstraction via Nonparametric Bayesian Inference. In *European Conference on Computer Vision (ECCV)*, pages 479–495. Springer, 2022.

[66] Zhijin Yang, Pengfei Xu, Hongbo Fu, and Hui Huang. WireRoom: model-guided explorative design of abstract wire art. *ACM Transactions on Graphics*, 40(4):1–13, 2021.

[67] Yunfan Ye, Renjiao Yi, Zhirui Gao, Chenyang Zhu, Zhiping Cai, and Kai Xu. NEF: Neural Edge Fields for 3D Parametric Curve Reconstruction from Multi-view Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8486–8495, 2023.

[68] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[69] Yibo Zhang, Lihong Wang, Changqing Zou, Tieru Wu, and Rui Ma. Diff3DS: Generating View-Consistent 3D Sketch via Differentiable Curve Rendering. *arXiv preprint arXiv:2405.15305*, 2024.

WIR3D: Visually-Informed and Geometry-Aware 3D Shape Abstraction

Supplementary Material

A. Data Preprocessing

We leverage the input surface not just in our SDF regularization, but also in generating supervision data specialized for our task. Specifically, we generate stylized Freestyle renders which isolate the key geometric features of a shape for our stage I optimization. In the case where a user does not supply keypoints, we leverage the priors of 2D foundation models to automatically detect keypoints which correspond to salient shape features.

Freestyle Rendering Standard opaque surface renders are not ideal for our curve representation, which are non-occlusive by construction. Optimizing with these surface renders can result in under-detailed abstractions or particular Janusing artifacts [52], where curves positioned on the opposite side of the viewed surface end up being optimized for the wrong side. Furthermore, when the shape is untextured, surface renders may be poor at exhibiting key geometric structures.

To resolve this, we render the shapes in a stylized fashion to allow for each view to isolate the shape geometric structure and take into account the occluded shape features. Specifically, we render the shapes using the Freestyle rendering engine [11] in Blender and render the shape in terms of its view-dependent contours, *without* accounting for occlusions. These Freestyle renders are purely based on the shape geometry and do not take into account any textures. Thus, these renders are appropriate for the first stage of our optimization (Sec. 3.3) where we focus on capturing the shape geometry.

Keypoint Detection. When keypoints are not included in the input, we automatically identify keypoints of interest on the shape's surface using the 3D feature extraction method developed in Backto3D [64]. This method back-projects 2D image features to a 3D shape using a simple averaging scheme. Our assumption, following Backto3D, is that these backprojected features contain meaningful information of the shape's salient visual features, and thus can be leveraged for identifying keypoints relevant to those features.

Specifically, we render views of the 3D model and encode them using CLIP RN50x16, back-project the pixel-level latent features to shape vertices, and average the features among duplicate vertices captured in different views.

Once we have 3D features on the shape, we apply KMeans clustering over these features [35] and obtain k latent clusters, where k is the number of keypoints we wish to obtain. We interpret these clusters as aggregating surface points with similar visual content. We identify the vertex whose features are closest to the cluster centers as keypoints, since these vertices are most likely to represent the key visual feature associated with the cluster. We make k the number of curves we initialize in stage 2 of our optimization, though this number can be adjusted depending the number of salient elements on the shape.

Figure 10. **NEF qualitative comparison.** We show NEF results on the same models we compare to 3Doodle in the main paper. NEF is specialized for simple manufactured CAD shapes, so it struggles to fit edges to more complex surfaces. This limitation was similarly observed in 3Doodle.

B. Neural Edge Field Comparison

We show a qualitative comparison to NEF in Fig. 10, using the same models we show in the main paper for 3Doodle, except for the models which NEF fails to produce meaningful point clouds for. Though NEF can capture the rough silhouette of the target shape, the method is specialized for simple manufactured surfaces with sharp corners, so it struggles to place curves meaningfully on more complex surfaces. This results in a messier and harder-to-identify abstraction.

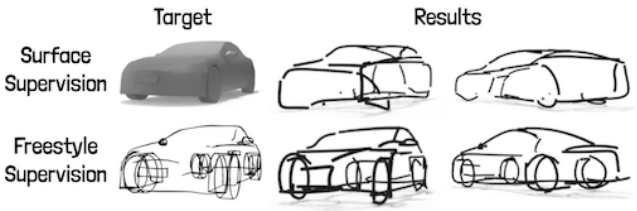

Figure 11. **Freestyle render ablation.** Running our method without freestyle renders still produces a reasonable abstraction, but key geometric features, such as the wheels of the car, may be missed due to the lack of visual content from the surface renders.

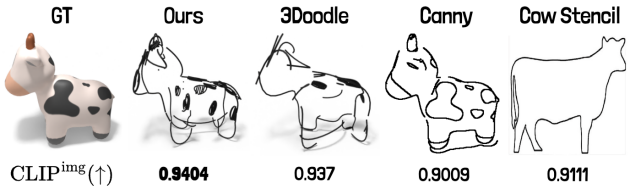

Figure 12. **Perceptual metrics reliability.** We show the unreliability of CLIP$^{\text{img}}$ in evaluating semantic similarity of curve abstraction to a target. We show for a given view, our stroke abstraction, 3Doodle's, the edge map for the view extracted using Canny edge detection [3], and a random image of a cow stencil obtained from Google. Note that though the Canny edge map captures the entire geometric structure and textures of the shape, its CLIP$^{\text{img}}$ score is shockingly lower than that of the stencil image. The vast difference in the two images also demonstrates how small differences in score can indicate major differences in quality.

C. Perceptual Metric Details.

The LPIPS metric is based on an AlexNet architecture trained for image classification fine-tuned with a linear layer on an annotated perceptual similarity dataset. Notably, we use the VGG variant of LPIPS for optimization, which is a commonly performed split between optimization and evaluation, and is similarly done in 3Doodle.

CLIP$^{\text{img}}$ is computed by encoding both the stroke renders and shape renders through a CLIP ViT/B-32 model, computing the cosine similarity, and scaling the score to [0-1]. Note that we only use the ResNet variants of CLIP for our optimization.

D. Ablations

Freestyle renders. We ablate on Freestyle render supervision in Fig. 11, instead running our method using opaque surface renders. The resulting abstraction is reasonable, but misses important geometric feature detail in the wheels and side mirrors of the car.

SDF loss. We ablate on the SDF loss in Fig. 19. The SDF loss prevents texture features from floating off the surface

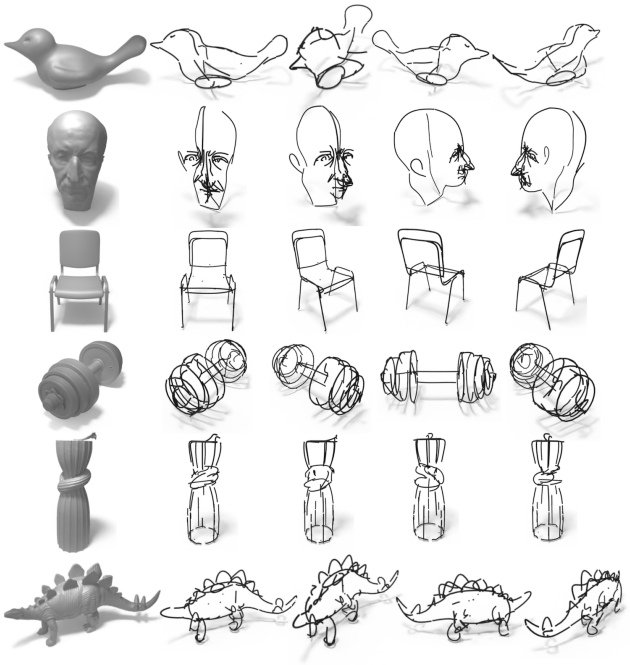

Figure 13. **Qualitative results for untextured shapes.** We show the result of our method on a collection of untextured meshes. Our method is effective and robust on a wide collection of different geometries.

implied by the rest of the strokes, such as the spots on Bob circled in red.

Random Keypoints Ablation. We ablate on the keypoint selection by running our method with the localized keypoint loss weighted by random keypoints sampled from the surface and compare the results to our method run without keypoint weighting (unweighted semantic loss). The results are in Fig. 18. Note that the quality of our method with random keypoints is the same as running our method with no keypoints, demonstrating our robustness to non-meaningful keypoints. Our localized keypoint loss is strictly better than the unweighted semantic loss when the keypoints are meaningful, and on par otherwise.

E. Additional Applications

Detail Refinement. We show an additional example of keypoint-based abstraction refinement in Fig. 20. For our refinement application, we freeze the existing curve set and optimize 6 new curves randomly initialized in a local Gaussian around each keypoint. We use the same losses as the main method and only sample views where the keypoints are visible, and optimize for 300 iterations.

Figure 14. **Multi-view fidelity.** WIR3D adheres to the abstracted object in a 3D-consistent manner such that its properties can be perceived from every viewing angle.

Curve-Based Shape Deformation. Our deformation application exploits the close correspondence between the optimized curves and key visual features on the input surface, thanks to the SDF and keypoint localization losses. We develop a simple skinning system for the surface where each vertex is assigned a set of skinning weights to points sampled on all the curves in the scene. These skinning weights are based on the L2 distance between each vertex and sampled point, and a softmax is applied to ensure they sum to 1. Transformations to each curve can then be automatically mapped to the surface through these skinning weights, and the procedure can be performed at interactive speeds. We implement this deformation system as a proof-of-concept script, and show videos of the working system in the supplemental material, with screenshots displayed in Fig. 8. Note that *no smoothing postprocesses* are applied to the mapped transformations, and the smoothness of the deformations

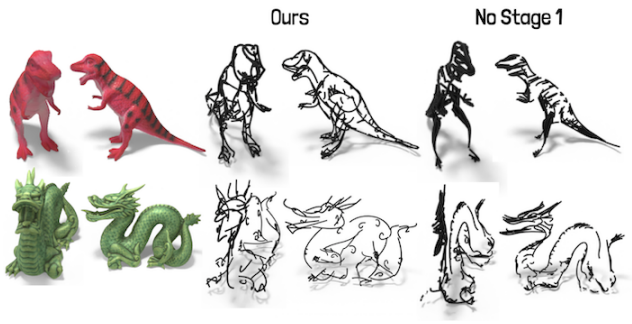

Figure 15. **Stage 1 ablation.** Stage 1 optimization is essential for capturing the full extent of the input geometry. Without it, the optimization tends to bias towards certain views, while the overall abstraction experiences a "flattening" effect.

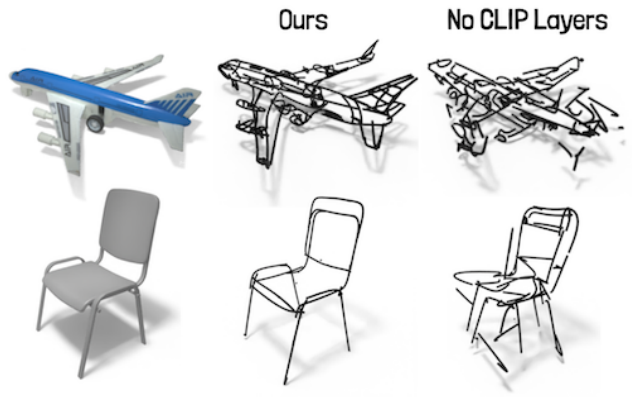

Figure 16. **CLIP layers ablation.** Supervising with the intermediate activations of CLIP is critical for maintaining coherent geometry. Using only the fully-connected CLIP output results in rough semantic abstraction, but the input shape geometric features are not well-preserved.

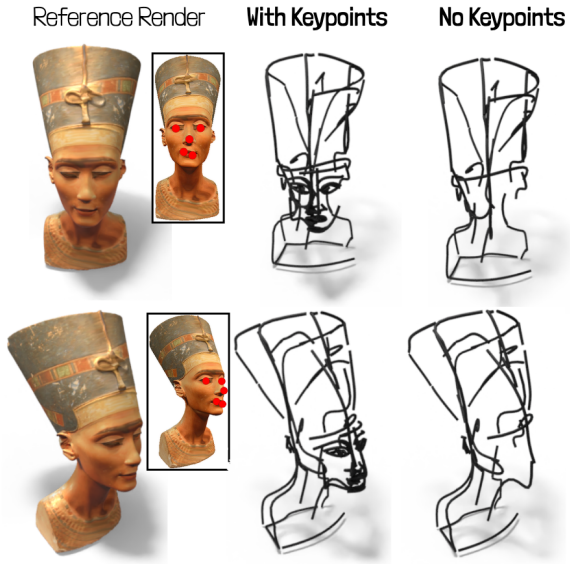

Figure 17. **Localized keypoint ablation.** We show the influence of our localized weighting framework when keypoints pinpoint semantically meaningful features. The result without keypoints is reasonable, but the face texture details are only captured with the introduction of keypoints for spatial weighting.

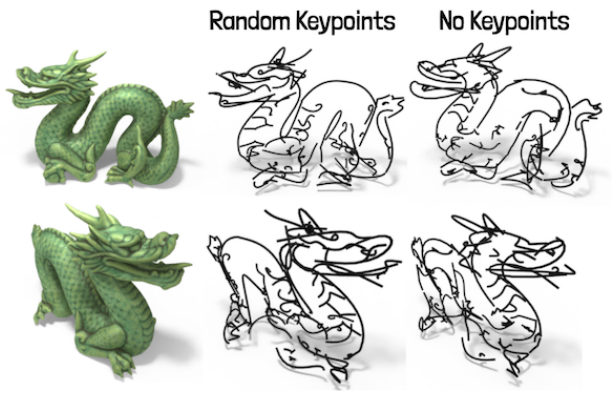

Figure 18. **Random keypoints ablation.** We compare our method with randomly sampled keypoints to our method with no keypoints (unweighted semantic loss). Note that the result with random keypoints is of similar quality to our method without keypoints, which demonstrates that our method is robust to non-meaningful keypoints.

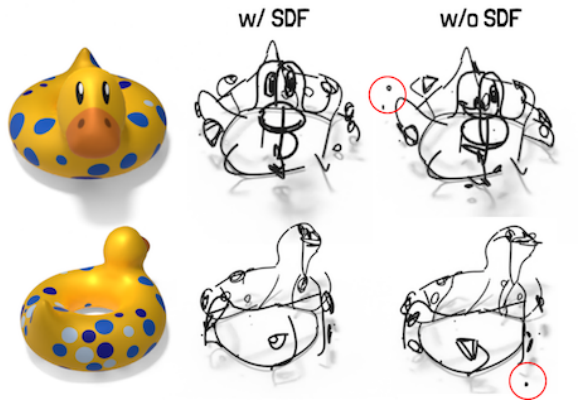

Figure 19. **SDF ablation.** The SDF loss helps to ensure abstracted visual features will stay anchored to the surface implied by the strokes. Without it, some features may hover outside the surface, such as the smaller spots on Bob.

Figure 20. **Texture keypoint control.** We expand on the keypoint control example shown in the main paper with a textured example. We show how by selecting keypoints on the texture on the plane, we are able to refine the abstraction by incorporating those texture elements.

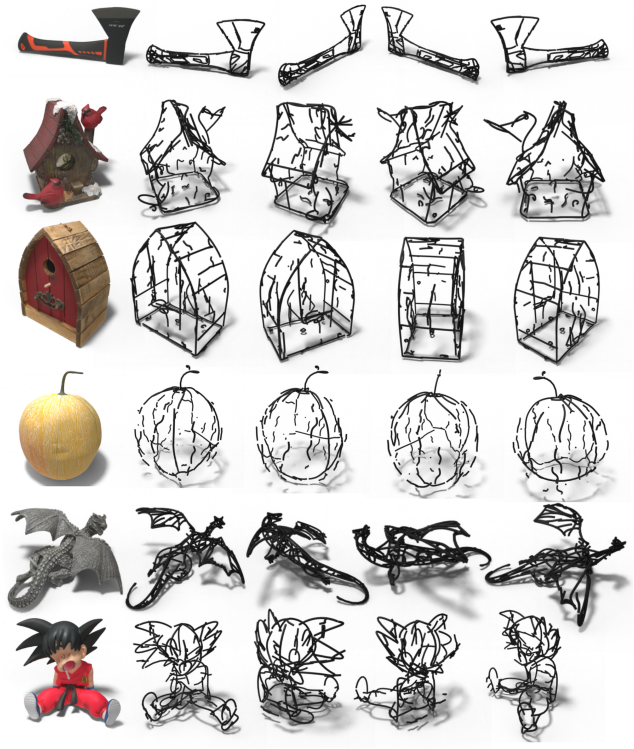

Figure 21. **Additional textured results.** We show additional textured results from the Meta DTC dataset [40].

are a result of the effectiveness of the curves in interpolating the quantities along the surface.

F. Additional Abstraction Results

Texture Abstractions. Additional results on textured shapes are shown in Fig. 21. Our method is robust to many different types of models ranging from manufactured shapes with sharp edges to organic shapes with complex curvature.

Scene Abstraction. We show an example of our method run on a large scene in Fig. 22. Our method is able to reproduce the global scene layout, and successfully abstracts objects at different scales in the scene (e.g. house, trees, animals).

G. Optimization Details.

For both stages, we optimize for 20000 iterations, sample 1 view per iteration, and use an ADAM optimizer with a learning rate of 1e-3. For CLIP supervision we sample 4 augmentations per view. In stage 1 of the optimization, we use the RN101 CLIP architecture, with $\lambda_{\text{fc}} = 0.1$. In stage 2, we use the RN50x16 architecture, $\lambda_{\text{lpips}} = 0.1$, $\lambda_{\text{fc}} = 75$, and $\sigma = 0.1$.

Figure 22. **Scene abstraction.** Our method extends to scene abstraction. Note how our method reproduces the global scene layout and captures all the objects in the scene despite the large scale differences.

H. User Study Screenshots

We show screenshots from our user study in Fig. 23. The question order and the order of Wir3D versus 3Doodle assignment to "Sketch1/Sketch2" are randomized for each respondent. All results shown are rotating gifs, so that users can evaluate based on the full 360 views of the abstraction. At the beginning of the study, we present three examples of abstractions of different quality and explain the factors that determine their quality, so that users can make more precise judgments in their visual evaluation.

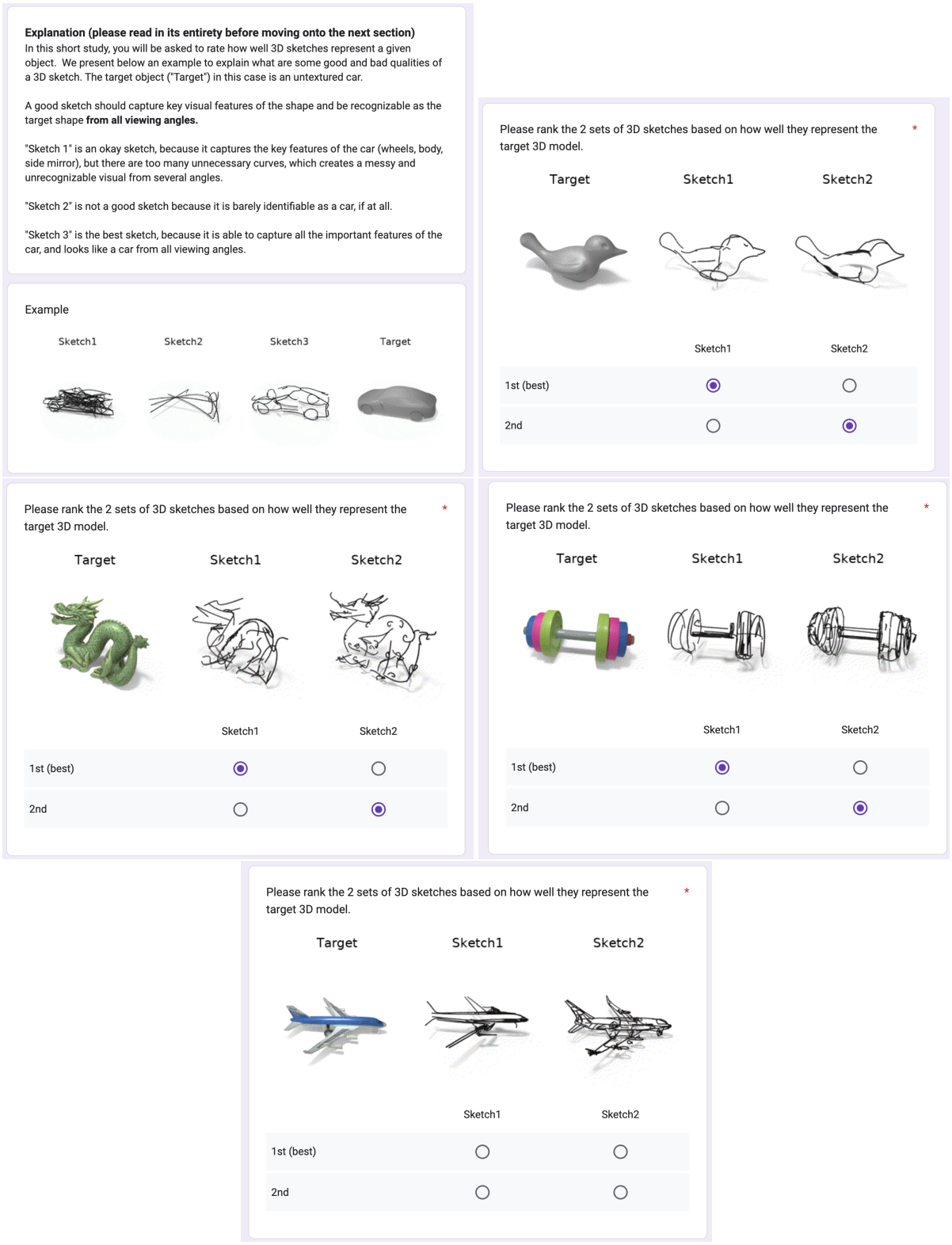

Figure 23. **Perceptual Study Screenshots.** Screenshots from our perceptual study. The question order and the order of Wir3D versus 3Doodle assignment to "Sketch1/Sketch2" are randomized for each respondent.

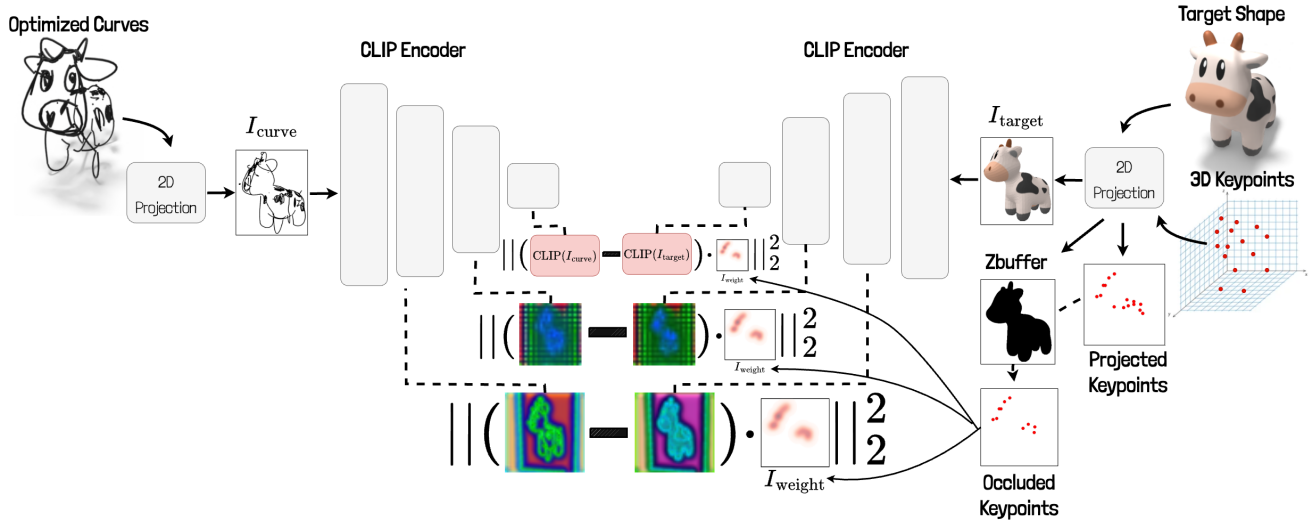

Figure 24. **Comprehensive Localized Keypoint Loss.** We show a comprehensive visualization of the localized keypoint loss.