arXiv:2604.13549v3 [cs.CV] 9 Sep 2026

Reconstruction of a 3D wireframe from a single line drawing via generative depth estimation

Elton Cao Hod Lipson

Creative Machines Lab, Columbia University
New York, NY

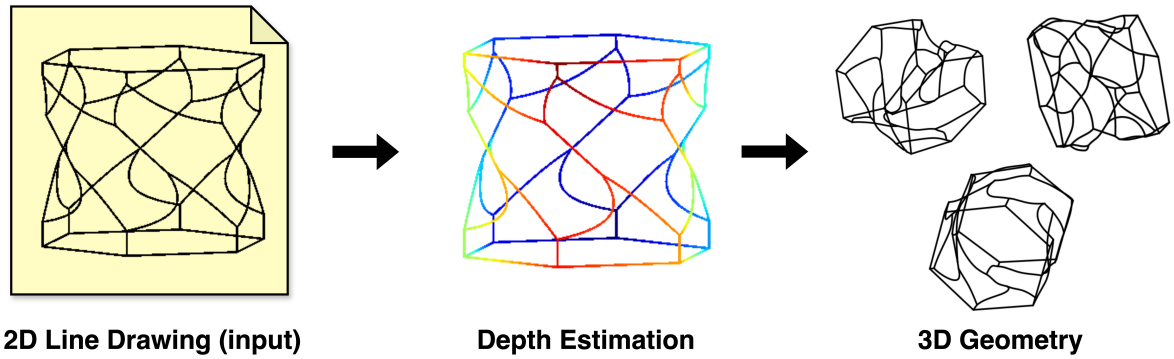


Figure 1. **The engineering line drawing reconstruction challenge.** A single orthographic line drawing presents numerous "possible" reconstructions, some more plausible than others. Our method predicts depth only on the drawn pixels, producing a projection-faithful 3D wireframe rather than completing unseen geometry or inferring a full CAD program.

Abstract

Reconstructing 3D geometry from 2D engineering line drawings is an inherently ambiguous problem: while visible strokes determine the object's projected structure, they do not specify the depth of each stroke. Rather than treating this problem as sketch-based asset generation, where models often infer unobserved structure, we study projection-faithful wireframe reconstruction: lifting a user-provided drawing into 3D according to its visible strokes. We formulate this task as conditional depth estimation over line drawings, predicting a depth value for each drawn pixel to produce a 3D wireframe. To model the ambiguities of orthographic projection, we implement a Latent Diffusion Model with spatial conditioning on the input sketch and optional partial-depth conditioning for iterative reconstruction. We train and evaluate our models on over three million synthetic image-depth pairs derived from CAD wireframes, including a newly curated corpus of roughly 90,000 shapes. Across varying shape complexities, our framework achieves robust reconstruction performance; scaling from 256 *to* 512 *resolution with a retrained latent space roughly halves reconstruction error, reaching a 3.9% best-of-five (7.0% average) normalized depth error. These results demonstrate the potential of projection-faithful depth estimation as a user-controlled approach for iterative 3D wireframe creation in engineering design.*

1. Introduction

Sketching is a natural and efficient medium for communicating geometric ideas, especially during the early stages of engineering design [9, 31, 45]. With a small number of strokes, a designer can express object structure, spatial relationships, and intended form before the object is fully specified in a formal modeling system [31, 43]. Converting such 2D drawings into 3D geometry, however, remains a long-standing challenge: a line drawing constrains the projected structure of an object, but generally does not uniquely determine the depth, topology, or spatial ordering of its strokes [13, 24, 29, 42]. This ambiguity is especially pronounced for engineering line drawings, which are sparse, unshaded, and dominated by visible edges, contours, and wireframe-like structures rather than the photometric cues

available in natural images [20, 35, 36, 50].

Sketch-based interfaces for modeling (SBIM) have addressed this problem through a wide range of interaction and reconstruction paradigms, including gesture-based primitive creation, contour inflation, extrusion, surface sketching, and multi-view modeling [14, 16, 30, 31, 52]. More recently, deep sketch-based 3D modeling (DS-3DM) has expanded this design space with neural networks, transformers, implicit representations, differentiable rendering, diffusion models, and foundation-model optimization [3, 5, 11, 21, 22, 26, 45, 48, 55–57]. These methods differ substantially in their input assumptions, viewpoint constraints, sketch styles, output representations, and evaluation metrics [45]. As a result, "sketch-to-3D" has emerged as not a single benchmark problem, but a family of related tasks with distinct output contracts.

We use output contract to refer to the representation a method is designed to produce, such as CAD command sequences, face graphs, watertight meshes, implicit fields, point clouds, textured assets, or depth values over visible sketch support. This distinction is central to engineering line drawing reconstruction. Many recent sketch-conditioned 3D methods use a sketch as a prompt for generating a complete semantic object, often inferring unseen structure, parts, textures, or appearance [2, 3, 10, 11, 26, 45, 55, 56]. Such methods are powerful for asset generation and conceptual modeling. In contrast, engineering drawing reconstruction often requires a projection-faithful representation: the system should preserve the geometry explicitly drawn by the user [21, 22, 24, 46].

In this paper, we study projection-faithful 3D reconstruction of engineering line drawings. Given a sparse 2D drawing of a designed object, our goal is to preserve the visible stroke support and infer depth values that embed the drawing as a 3D wireframe, allowing a user to "draw in 3D". Thus, the resulting model predicts depth for drawn pixels rather than producing a CAD program, face graph, or completed semantic asset. This formulation provides a direct route from 2D to 3D for engineering line drawings while avoiding assumptions that the input must correspond to a complete CAD command sequence, a watertight surface, or a valid set of closed faces.

Alternatively, existing engineering-sketch reconstruction systems impose more restrictive assumptions. CAD-command approaches such as Sketch2CAD, DeepCAD, and Free2CAD represent geometry as sequences of parametric operations, stroke groupings, or sketch-extrude programs, enabling editability and structure but constraining the output to a predefined vocabulary [21, 22, 48]. Topological methods such as Neural Face Identification reconstruct face connectivity from wireframe projections, but require the drawing to support valid face loops or manifold-style structure [46].

We formulate engineering line drawing reconstruction as conditional depth estimation over the visible sketch support. If a depth value is predicted for each drawn pixel, the 2D drawing can be lifted directly into an embedded 3D wireframe while preserving the input projection. This image-based formulation aligns with SBIM systems that represent strokes as raster images for compatibility with image-processing pipelines [31], and it avoids requiring vectorization, stroke grouping, primitive fitting, or face identification as a prerequisite for reconstruction [21, 22, 24, 46]. Because orthographic line drawings lack shading, texture, perspective, and natural-image context, multiple 3D depth assignments may remain plausible for the same 2D support [13, 29, 31, 42]. We therefore model the task generatively using a Latent Diffusion Model (LDM) conditioned on the input drawing and, when available, partial-depth information from previous reconstruction steps [12, 17, 37].

The partial-depth conditioning supports an iterative sketch-reconstruct-sketch workflow. A user can draw an initial set of strokes, reconstruct them in 3D, and then add new strokes conditioned on the existing 3D structure. This follows the progressive modeling goals of SBIM while using depth-space geometric context rather than CAD-command context [21, 22, 31]. Thus, the result is a projection-faithful mechanism for lifting sparse engineering line drawings into 3D without requiring a full CAD kernel, feature-history tree, or semantic 3D generator.

Our contributions are as follows:

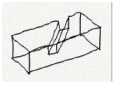
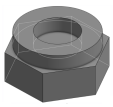
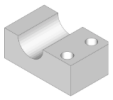
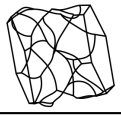
- A projection-faithful formulation of engineering line-drawing reconstruction as conditional depth estimation on visible sketch support.
- A diffusion-based reconstruction model with spatial sketch conditioning and partial-depth conditioning for iterative sketching.
- Two large-scale rendered datasets derived from ABC geometry: over one million wireframe/depth pairs from a filtered prior-work subset, and a newly curated curve-rich corpus of roughly 90,000 shapes yielding 2.1 million pairs.

2. Related Work

2.1. Sketch-Based Modeling and Deep Sketch-Based 3D Modeling

Sketch-based interfaces for modeling have a long history in computer graphics, geometric modeling, and human-computer interaction [14, 31, 43, 52]. Olsen et al. organize SBIM systems by how sketches are interpreted. This includes instructions to create new 3D models, augment existing models, or deform and manipulate geometry [31]. This highlights that sketch interpretation depends entirely on the intended output representation and interaction mode. Engineering-style drawings have historically motivated fitting, beautification, constraint inference, and geometric optimization because precision and structural integrity are cen-

Table 1. **Most closely related work for engineering line drawing reconstruction.** We compare methods that reconstruct 3D structure from line drawings, wireframes, or CAD-style sketches of designed objects. Because these methods produce different output representations, direct metric-to-metric comparison is difficult. The table emphasizes each method's output contract and assumptions. The "Shape" column shows a reconstruction among the most complex demonstrated in each publication (images reproduced from the cited papers); for ours, the cell shows an actual reconstruction by our model of a 60-stroke test sketch in which 34 strokes are free-form B-spline curves.

Reference	Input domain	Technique	Key assumptions / limitations	Output contract	Shape	Code
Huffman (1971) [13]	Polyhedral line drawings	Qualitative line labeling for concave, convex, and occluding edges.	Determines symbolic consistency, but does not recover a 3D shape.	Line labels	—	N/A
Sugihara (1986) [42]	Polyhedral line drawings	Geometric analysis of whether a line drawing can correspond to a valid 3D object.	Studies constraints, but does not provide a learned reconstruction pipeline.	Geometric feasibility	—	N/A
Lipson and Shpitalni (1996) [24, 25]	Engineering line drawings / wireframes	Optimizes vertex depths using geometric regularities such as planarity, parallelism, orthogonality, and symmetry.	Requires clean vector graphs and reliable vertex-edge structure, but brittle for ambiguous sketches.	3D wireframe graph		N/A
Sketch2CAD (2020) [21]	CAD-style sketches	Parses sketches into sequential CAD modeling operations and parameters.	Produces editable CAD-like programs, but is constrained by a fixed vocabulary.	CAD command sequence		Open
DeepCAD (2021) [48]	CAD construction sequences	Learns a generative model over CAD command sequences.	Models CAD programs rather than arbitrary input line drawings & constrained by a fixed vocabulary.	CAD command sequence		Open
Free2CAD (2022) [22]	Freehand CAD sketches	Groups strokes into geometric compounds and fits CAD operations.	Handles rougher sketches than command-only methods, but still constrained by a fixed vocabulary.	CAD operations		Open
Neural Face Identification (2022) [46]	CAD-derived wireframes	Predicts face loops and topological connectivity from 2D wireframe drawings.	Assumes the drawing supports valid closed faces and face connectivity.	Face graph		Open
Ours	Engineering line drawings / CAD-derived wireframes	Predicts depth values over visible sketch support using conditional generative depth estimation.	Does not recover a CAD command sequence, explicit face topology, or watertight surface.	Projection-faithful 3D wireframe depth		Open

tral to the task [24, 31, 43].

Recent DS-3DM methods extend this literature with learning-based models and large paired datasets [45]. Tono et al. categorize modern methods according to input amount, viewpoint assumptions, sketch style, model architecture, part structure, output options, geometry type, and evaluation metrics [45]. Their survey includes neural models, implicit representations, transformers, diffusion models, differentiable rendering, and foundation-model optimization [2, 3, 5, 11, 26, 45, 55–57]. This literature provides the broader context for our work, while also motivating a narrower comparison set: methods that produce textured assets, NeRFs, Gaussian splats, implicit fields, point clouds, CAD operations, and face graphs solve related but distinct output contracts.

2.2. Classical Line-Drawing Interpretation

3D reconstruction from line drawings was studied before modern learning-based modeling as a problem of symbolic interpretation, geometric feasibility, and constraint satisfaction [4, 13, 24, 42]. Huffman and Clowes introduced line-labeling frameworks for interpreting polyhedral scenes, while Sugihara studied the consistency and realizability of line drawings under geometric constraints [4, 13, 42]. These works established a central challenge for line-drawing reconstruction: a 2D drawing may admit multiple valid 3D interpretations, and some drawings may not correspond to a physically realizable object at all [13, 31, 42].

Metric reconstruction methods introduced geometric priors to recover plausible 3D structure. Lipson and Shpitalni formulated line-drawing reconstruction as an optimization

problem over vertex depths using regularities such as planarity, parallelism, perpendicularity, symmetry, and isometry [24, 25]. Masry and Lipson later explored neural and evolutionary approaches for reconstructing 3D wireframes from 2D sketches, showing that learned priors could complement geometric regularity constraints [28]. These works are historically close to ours because they address line-drawing-to-wireframe reconstruction rather than semantic asset generation, serving as an older lineage work to our work.

2.3. Engineering Sketch and CAD Reconstruction

A major branch of recent work reconstructs engineering drawings by mapping sketches into CAD-like or topology-aware representations. Sketch2CAD treats sketch interpretation as a sequence of CAD modeling operations, using existing partial geometry to decode later sketch commands [21]. DeepCAD learns a generative model over CAD command sequences, demonstrating that CAD construction histories can be represented as learnable sequential data [48]. Free2CAD extends this direction to freehand CAD sketches by grouping strokes into geometric compounds and fitting parametric operations with transformer-based architectures [22, 45]. Related program- and parameter-based systems such as GeoCode, PICASSO, and SECAD-Net further demonstrate the value of symbolic or procedural representations for editable 3D design [15, 23, 34].

These methods are highly relevant because they target engineering-style inputs. Their strength is editability: by recovering CAD operations or parametric programs, they produce representations that can be manipulated in downstream modeling systems. Thus, their assumptions maintain that strength. The object must be expressible in the chosen vocabulary of primitives, operations, or sketch-extrude programs [21, 22, 48]. Our method makes a different tradeoff by predicting depth on visible sketch support, sacrificing immediate CAD editability but reducing dependence on primitive vocabulary and other restraints.

Topological reconstruction methods provide another close comparison. Neural Face Identification predicts face loops from 2D wireframe projections of manifold objects, framing wireframe reconstruction as a sequence prediction problem over graph connectivity [46]. Its output contract is face topology, whereas ours is depth over visible sketch pixels. The two formulations are complementary, since a face-topology model can recover surface structure when valid face loops are present, while depth lifting can operate on partial, open, or intermediate wireframes that may not yet define closed faces.

2.4. Domain-Specific Engineering Sketch Reconstruction

Several recent systems restrict the domain or drawing convention to make sketch-based reconstruction tractable. Sketch2PQ reconstructs planar quadrilateral meshes from axonometric architectural sketches by assuming roof-like PQ mesh structure and using explicit curve categories [6]. Vitruvio reconstructs building geometry from single perspective sketches, adapting single-view reconstruction ideas to architecture-specific data [44, 45]. These methods show that engineering-specific assumptions can be powerful when the drawing convention and output representation are fixed.

2.5. Sketch-Based 3D Asset Generation

Many modern sketch-based 3D methods focus on object completion or asset generation. Sketch2Mesh, Sketch2Model, DeepSketch, LAS-Diffusion, Control3D, SketchDream, Sketch2NeRF, and Sketch3D map sketches to meshes, implicit fields, NeRFs, Gaussian splats, point clouds, or textured 3D assets [2, 3, 11, 26, 45, 54–57]. These methods often use learned object priors, differentiable rendering, text prompts, foundation models, or diffusion losses to infer geometry and appearance not explicitly present in the input sketch [2, 3, 26, 55]. This design point is well suited to creative asset generation, where a sketch may serve as an incomplete prompt for a full 3D object.

Our work occupies a different design point in the DS-3DM taxonomy. The goal is not to complete a semantic object, synthesize appearance, or infer hidden parts, but to preserve visible stroke support and estimate depth values for those strokes. Broad sketch-to-asset systems are therefore important context, but they are not direct baselines for our task.

2.6. Depth Estimation and Generative Depth Priors

Monocular depth estimation has traditionally been studied on natural images, where models can exploit shading, texture, perspective, object semantics, and large-scale image priors [8, 35, 36, 50]. Recent models such as MiDaS, DPT, Depth Anything, and Depth Anything V2 achieve strong generalization across photographic datasets, but their assumptions do not directly transfer to sparse engineering wireframes [35, 36, 50, 51]. Engineering line drawings lack the photometric and semantic cues that make natural-image depth estimation effective.

Generative depth models provide a natural framework for ambiguous reconstruction. Diffusion models have achieved strong results in image generation and conditional synthesis by modeling multimodal output distributions rather than regressing a single deterministic target [12, 37, 41]. Marigold adapts latent diffusion priors to monocular depth estimation, demonstrating that diffusion-based models can be effective for ill-posed depth prediction when trained or fine-tuned for the appropriate domain [17]. Our work adopts this generative perspective for engineering line drawings.

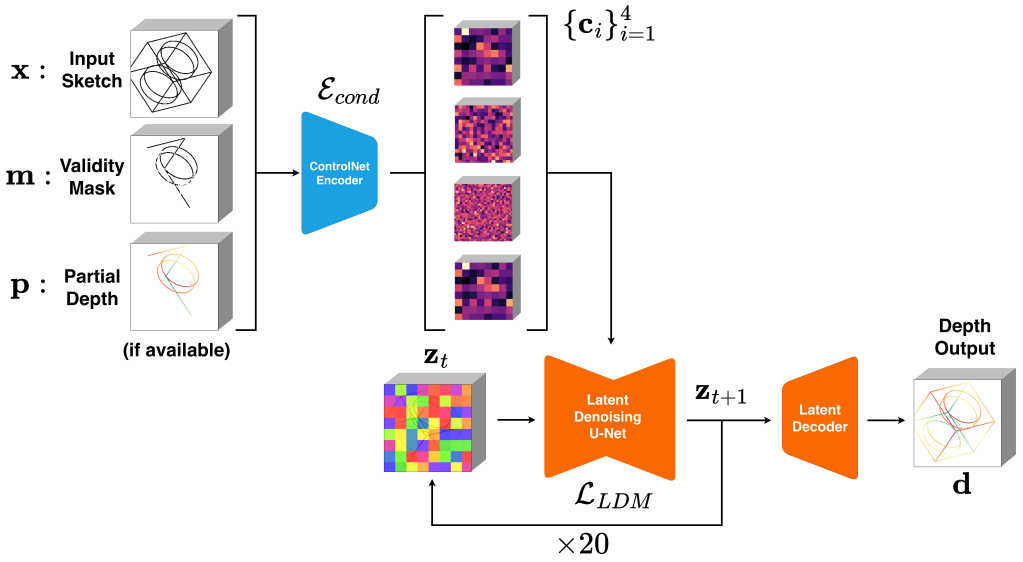


Figure 2. **LDM architecture with ControlNet conditioning.** Our model's diffusion architecture involves passing the geometric conditions ($\mathbf{x}$, $\mathbf{p}$, and $\mathbf{m}$) into our conditioning encoder, outputting representations at various resolutions, $\{\mathbf{c}_i\}_{i=1}^4$. These representations are then injected into the LDM, which predicts the Gaussian noise added to the latent space depth maps. The resulting latent space is then decoded into the depth map output.

3. Problem Formulation

We frame the task of wireframe reconstruction as a conditional dense depth estimation problem. Unlike traditional CAD reconstruction, which aims to recover a sequence of parameters or a graph of discrete geometric primitives, our goal is to learn a mapping from a 2D raster sketch to an embedded 3D wireframe representation via per-pixel depth.

Let $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ represent a 2D raster sketch of a wireframe object. This input $\mathbf{x}$ is assumed to be a projection of an underlying 3D ground truth geometry $\mathcal{G}$. Our objective is to recover the depth map $\mathbf{y} \in \mathbb{R}^{H \times W}$ corresponding to the camera viewpoint of the projection $\mathcal{G}$. The value $\mathbf{y}_{u,v}$ represents the normalized distance from the camera plane to the geometry at pixel coordinates (u, v).

The fundamental challenge in single-view wireframe reconstruction is that the mapping from $\mathbf{x}$ to $\mathbf{y}$ is ill-posed [13, 29, 31, 42]. Under orthographic projection, visual cues such as perspective foreshortening are absent. Consequently, a single 2D wireframe $\mathbf{x}$ may mathematically correspond to multiple valid 3D interpretations (e.g., the Necker Cube ambiguity). Because a deterministic regression model would tend to average these possibilities, we instead formulate the problem as learning the conditional probability distribution:

$$p(\mathbf{y}|\mathbf{x})$$

This probabilistic formulation allows the model to generate a specific and valid 3D structure by sampling from the learned distribution of valid geometries.

To support the "sketch-reconstruct-sketch" workflow, we introduce a secondary conditioning variable, the partial depth map $\mathbf{p}$ and a validity mask $\mathbf{m}$. These variables represent sparse or incomplete 3D information (e.g., existing geometry from a previous iteration of the design). The input to our model is thus a combination of $(\mathbf{x}, \mathbf{p}, \mathbf{m})$, where:

- $\mathbf{x}$ is the current binary input sketch mask.
- $\mathbf{p}$ is the existing/partial depth.
- $\mathbf{m}$ is the validity mask indicating which partial depth pixels can be fully trusted.

The full objective is to model the conditional distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{p}, \mathbf{m})$.

4. Projection-Faithful Reconstruction via Generative Depth Estimation

Following widely successful diffusion-based approaches [12], we model the conditional posterior distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{p}, \mathbf{m})$ using a Latent Diffusion Model (LDM) designed for high-resolution image synthesis [37].

4.1. Latent Diffusion Framework

To maintain computational efficiency while preserving fine-grained geometric details, we perform the diffusion process in a compressed latent space. We utilize a variational autoencoder (VAE) [18] consisting of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. The target depth map $\mathbf{y}$ is encoded into a latent rep-

resentation $\mathbf{z}_0 = \mathcal{E}(\mathbf{y})$, where $\mathbf{z}_0 \in \mathbb{R}^{h \times w \times c}$ is spatially downsampled by a factor of $f = 8$.

The diffusion process is modeled as a Markov chain that incrementally destroys the structure of $\mathbf{z}_0$ by adding Gaussian noise over T timesteps, resulting in a sequence $\mathbf{z}_1, \ldots, \mathbf{z}_T$, where $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$.

The generative process is learned by training a time-conditional denoising network ϵ_θ to predict the noise component ϵ added to a noisy latent $\mathbf{z}_t$, conditioned on the timestep t and our geometric conditions $\mathbf{c} = (\mathbf{x}, \mathbf{p}, \mathbf{m})$. The objective function is the standard noise-prediction loss:

$$\mathcal{L}_{LDM} = \mathbb{E}_{t, \mathbf{z}_0, \epsilon, \mathbf{c}} \left[\|\epsilon - \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c})\|_2^2 \right]$$

4.2. Network Architecture and Conditioning

Our denoising backbone ϵ_θ follows the U-Net architecture [38] used in Stable Diffusion, featuring alternating ResNet blocks and spatial self-attention mechanisms.

ControlNet-Style Injection. Unlike semantic text conditioning, which is typically handled via cross-attention, sketch conditioning requires strict spatial alignment between the input strokes and the output geometry. To enforce this pixel-to-pixel correspondence, we adopt a ControlNet-style conditioning mechanism [37, 53]. The ControlNet encoder extracts a set of multi-scale conditional feature maps, collectively denoted as $\{\mathbf{c}_i\}_{i=1}^4$, at various spatial resolutions. These hierarchical representations are then added directly to the corresponding intermediate hidden states of the U-Net decoder (Figure 2).

4.3. Conditioning Encoder Variants

The choice of $\mathcal{E}_{cond}$ is critical for interpreting the sparse, abstract nature of wireframe drawings. Considering the domain shift of line drawings compared to natural images, we investigate three distinct architectures for the conditioning branch:

1. **Latent Encoder (VAE-KL):** We initialize $\mathcal{E}_{cond}$ using the weights of the primary VAE encoder, representing an encoder pre-trained specifically to understand underlying wireframe geometries.
2. **Vision Transformer (ViT):** We employ a standard ViT architecture trained from scratch [7].
3. **Pre-trained DinoV2:** We leverage a ViT pre-trained with the DinoV2 objective [32]. Despite being trained on natural images, DinoV2 encodes rich, generalized visual representations that surpass what a wireframe-only encoder can initially capture. Thus, we fine-tune this expansive prior specifically to the wireframe domain.

4.4. Geometric Representation: Fixed-Range Normalized Disparity

A critical design choice in our reconstruction pipeline is the representation of geometry. Rather than predicting metric depth directly, we predict a normalized disparity field derived from inverse depth. This choice compresses the dynamic range of the target and places all training examples into a consistent bounded interval.

Let $Z_{u,v}$ denote the rendered depth at pixel (u, v). For foreground pixels, we first convert depth to raw disparity via inverse depth, and then normalize it using fixed near and far depth planes, Z_{near} and Z_{far}. In our implementation, to satisfy global orthographic constraints, these are defined as global constants rather than object-specific bounds. The target is therefore defined as:

$$y_{u,v} = \frac{\frac{1}{Z_{u,v}} - \frac{1}{Z_{\text{far}}}}{\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}}}$$

5. Dataset Generation

To train a generative model capable of generalizing across diverse engineering wireframes, we construct large-scale datasets of wireframe-depth pairs derived from the ABC Dataset [19]. We generate two datasets. The first (v1) begins from the filtered ABC subset used by Wang et al. [46], which removes unnecessarily complex shapes and duplicates, resulting in 10,076 unique CAD models. We use this subset to maintain alignment with prior wireframe-specific reconstruction work and render each model from multiple orthographic viewpoints to produce over one million paired sketch and depth samples at 256×256 resolution. This dataset supports our conditioning-encoder and model-scaling study. The second, expanded dataset (v2) is curated directly from the raw ABC corpus with our own filter family (Section 5.2) and rendered at 512×512, producing 2,138,056 pairs across roughly 90,000 unique CAD models, resulting in a $9\times$ increase in shape diversity. This dataset trains our final high-resolution model.

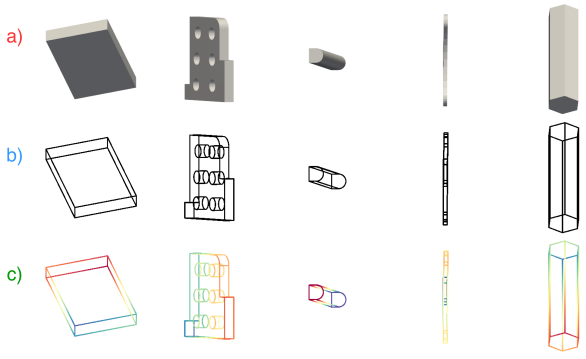


Figure 3. **Wireframe generation pipeline.** Our wireframe generation pipeline begins with a) the initial CAD object imported from ABC, into b) the CAD wireframe/sketch mask, and finally c) the resulting depth map of the sketch projection.

5.1. Rendering Pipeline

For each valid CAD, we render 2D wireframe projections and corresponding ground truth depth maps by modifying the rasterization pipeline in Wang et al. [46] built on PythonOCC (Figure 3).

- **Camera Model:** We utilize an orthographic projection camera and sample viewpoints uniformly from a hemisphere. In v1, each wireframe was rendered from 100 randomly sampled views. In v2, each wireframe is rendered from 24 random views, relying on shape diversity rather than view diversity. Both datasets are enhanced at training time by dataloader augmentation (Section 5.3).
- **Stroke Width:** v2 renders strokes at half the physical diameter of v1 while maintaining a raster width of roughly 2 pixels at 512×512. Thinner physical strokes increase the visual separation between adjacent edges, directly reducing the accidental pixel ratio identified as a primary error driver in our analysis.
- **Depth Maps:** Ground truth depth maps are generated via Z-buffering.

Example renders at each stage of the pipeline are shown in Figure 3.

5.2. Expanded Shape Curation

For the expanded dataset, we replace the fixed Wang et al. subset with a streaming curation pipeline over the raw ABC chunks: each chunk is downloaded, inventoried, filtered on shape metadata, deduplicated against previously accepted shapes, and rendered before moving to the next chunk. The filters target moderate complexity with geometric variety, allowing a single root solid, at most 96 edges and 192 faces, and caps on the short-edge fraction and normalized total edge length. We avoid penalizing curved geometry, as heavily curved and organic shapes are where depth-based reconstruction is most valuable.

This curation philosophy diverges from how prior engineering-sketch systems construct training data. CAD methods synthesize training corpora from sketch-and-extrude sequences whose profile vocabulary is limited to lines, arcs, and circles [22, 48]. Free-form B-spline geometry is unrepresentable in the data by construction. Topology-based methods filter ABC to shapes that satisfy their structural assumptions. The subset of Wang et al. [46], which our v1 dataset adopts, retains manifold shapes with valid face loops and modest complexity. Table 2 identifies the statistical differences: our expanded dataset doubles the structural complexity (median 36 vs. 18 edges per shape; maximum 96 vs. 53) while retaining curve-rich and free-form geometry that prior filtering excludes. The complexity and curves our benchmark includes are exactly those that prior training distributions exclude or under-represent.

Table 2. **Training-corpus comparison: prior-work ABC subset (v1) vs. our expanded curation (v2).** All statistics are computed with the same measurement code on both datasets. The expanded dataset doubles admissible structural complexity, extending into regimes excluded by prior filtering.

	v1 (Wang et al. subset)	**v2 (ours)**
CAD shapes	10,076	90,951
Rendered pairs	1.00M	2.14M
Resolution	256×256	512×512
Edges per shape (median / p90 / max)	18 / 39 / 53	36 / 77 / 96

5.3. Simulating Partial Depth

A key contribution of our method is the ability to reconstruct from partial depth. To train this capability, we must simulate realistic intermediate states of a drawing. An intermediate state has two components: which strokes the user has drawn so far, and which of those strokes already carry depth from a previous reconstruction step. We simulate both on the shared edge-vertex graph $\mathcal{G} = (V, E)$ of the visible wireframe.

For the depth component, naive random masking (e.g., dropping 50% of pixels) fails to capture human drawing behavior, as users typically draw connected strokes rather than scattered pixels. We instead propose a graph-based masking strategy:

1. For 50% of training samples, we select a random "start node" $v_{start} \in V$.
2. We perform a breadth-first search (BFS) traversal starting from v_{start}.
3. We retain the depth values for all edges visited by the BFS until a random threshold $k \sim U[10\%, 90\%]$ of the total wireframe is covered.

For the stroke component, we apply a similar shape-level augmentation on the same graph. A random subset of edges is removed entirely, simulating a drawing whose geometry is itself incomplete rather than merely lacking depth. In the expanded dataset, both mechanisms are applied in the dataloader so every epoch sees fresh intermediate states of every drawing.

5.4. Dataset Statistics

The v1 dataset consists of 1,004,051 unique pairs from 10,076 shapes, a magnitude increase over the approximately 10,076 samples used in previous wireframe-specific works [46]. The expanded v2 dataset consists of 2,138,056 unique pairs at 512×512 resolution from roughly 90,000 curated shapes, increasing shape diversity by another order of magnitude.

6. Experimental Setup

6.1. Implementation Details

We implement our framework using PyTorch [33] and the Hugging Face library [47], leveraging the latter to integrate pre-trained backbones (e.g., DinoV2) for the conditioning encoder. Models in the conditioning-encoder and scaling study are trained on the v1 dataset at a spatial resolution of 256×256 pixels. Our final model is trained on the expanded v2 dataset at 512×512.

We utilize the AdamW optimizer [27] for training. To preserve the learned features of pre-trained encoders (such as DinoV2 or the VAE-KL encoder), we apply a differential learning rate strategy: the learning rate for the pre-trained conditioning encoder is set one order of magnitude lower than that of the randomly initialized denoising U-Net.

Training of the v1 models was performed on a cluster of $4\times$ NVIDIA RTX 4090 GPUs, with a batch size of 192 for 100 epochs. Total training time was about 128 GPU hours for smaller models (VAE and small ViTs) and about 336 hours for the larger scale U-Nets with the DinoV2-Base configuration. For our largest v1 model, we upgraded to a $4\times$ NVIDIA RTX 5090 rig, with a total training time of 240 GPU hours.

Migrating to 512×512 required retraining both stages of the latent pipeline: the retrained VAE removes the reconstruction ceiling of the original autoencoder (Section 7.5), and because the latent space changes, the U-Net is retrained from scratch rather than fine-tuned. The model is trained on $4\times$ NVIDIA H100 GPUs at an effective batch size of 200 for 25 epochs (approximately 310 GPU hours).

6.2. Benchmarking Setup & Evaluation Metrics

Because our model outputs normalized depth over visible sketch pixels, we evaluate reconstruction quality with depth metrics. This output contract differs from CAD-command reconstruction, face-loop recovery, mesh completion, and sketch-to-asset generation, whose metrics typically measure command accuracy, topology recovery, Chamfer distance, IoU, or perceptual quality [21, 22, 45, 46]. Direct metric-to-metric comparison is therefore meaningful only when output spaces are compatible or can be converted without adding additional assumptions. Converting a CAD program to depth (or depth to a CAD program) scores one contract with the other's metric and favors whatever contract is being compared on. Thus, only against monocular depth estimators are we able to compare numerically (Table 4, Figure 5). Against engineering reconstruction systems with different contracts, we can only compare on assumption coverage: Table 3 records, from each method's own stated assumptions, which inputs it admits and which outputs it produces.

We evaluate on two held-out test suites, one per dataset. For the v1 study, we use the 504 shapes inside our dedicated test dataset and render four isometric viewpoints per shape, resulting in a benchmarking dataset of 2016 unique viewpoints. To quantitatively evaluate the geometric accuracy of the reconstructed depth maps, we report standard depth estimation metrics. To account for the stochastic nature of the diffusion process, we generate five predictions per sample, reporting both the average and the best-case performance. Let y_i denote the ground truth normalized disparity and $\hat{y}_i$ denote the predicted disparity at pixel i, for a total of N valid pixels.

1. **Mean Absolute Error (MAE):** This metric measures the raw magnitude of errors in a set of predictions in units of globally normalized disparity.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

2. **Normalized MAE (NMAE):** This metric normalizes the MAE to each image's disparity range. Most shapes would not fill the global disparity bound; thus, this metric is necessary for per image understanding.

$$\text{MAE}_{\text{normalized}} = \frac{\text{MAE}}{\max(y) - \min(y) + \epsilon}$$

3. **Absolute Relative Error (AbsRel):** This metric normalizes the error relative to the ground truth value, penalizing errors more heavily in regions of lower disparity (further depth).

$$\text{AbsRel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}$$

4. **Threshold Accuracy (δ):** We measure the percentage of pixels where the ratio between the predicted and ground truth values is within a specific threshold. We report the standard metric $\delta < 1.25$:

$$\delta = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\max\left(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right) < 1.25\right)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Depth-reversal-aware scoring. Under orthographic projection, every line drawing admits two equally valid depth interpretations related by reflecting all depths about the scene center. Thus, we therefore score each prediction against both the annotated ground truth and its depth-reversed counterpart, and report the metrics of the better-matching interpretation.

In total, we run experiments covering each conditioning encoder (VAE-KL, ViT small scratch, DinoV2). Then, we perform additional ablations studies on the DinoV2 model by halving the dataset size and testing the newer DinoV3 encoder [40]. Lastly, to determine how the model reacts to compute scaling, we provide DinoV2 and V3 base models with $3\times$ and $6\times$ larger variants.

Table 3. **Assumption comparison across output contracts.** Marks are derived from each method's stated input requirements and output representation (✓ supported, ∼ partially supported, × unsupported by the method's assumptions). Our benchmark corpus contains geometrically freeform edges expressible by neither a line nor a circular arc ("curves beyond fixed vocabulary") and is curated without manifold or closed-face requirements ("open/partial wireframes"). Depth Anything V2 shares our contract and is compared numerically in Table 4.

Method	Raster input (no stroke order / vectorization)	Curves beyond fixed vocabulary	Open / partial wireframes	Preserves drawn projection	Iterative partial-3D conditioning	Metric 3D output	Editable CAD output
Lipson & Shpitalni [24]	×	×	×	✓	×	✓	×
Sketch2CAD [21]	×	×	×	×	✓	✓	✓
DeepCAD [48]	×	×	×	×	×	✓	✓
Free2CAD [22]	×	∼	×	×	✓	✓	✓
Neural Face ID [46]	×	∼	×	✓	×	∼	×
Depth Anything V2 [51]	✓	✓	✓	✓	×	✓	×
Ours	✓	✓	✓	✓	✓	✓	×

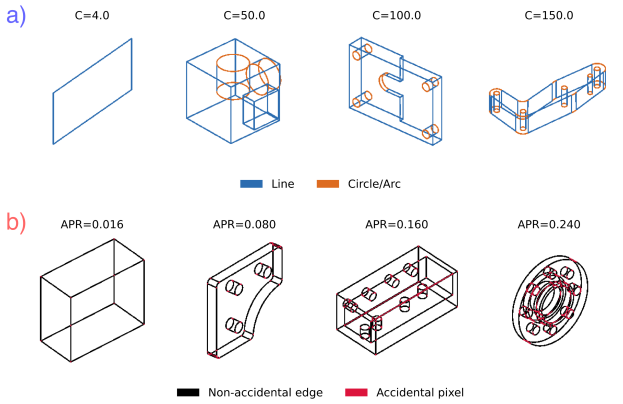


Figure 4. **Example shapes of various a) shape complexities and b) APRs.** To understand the effect of various measurements of shape complexity on model performance, we segment our dataset based on these parameters. For shape complexity, we color code by the primitive type, and for APR, we mark accidental pixel regions as red.

In addition, to gauge how the model performs against shapes of various complexity, we provide two additional parameters: accidental pixel ratio (APR) and curve complexity. APR is computed as the proportion of all foreground pixels that occlude one or more components, while curve complexity ratio is a summation of the per-edge primitive costs. To keep calculations simple and robust to potentially unclean CAD data, we assign a score of 1 to lines and a score of 3 to non-lines. Figure 4 displays example shapes of various shape complexities and APRs.

7. Results & Discussion

7.1. Comparison with Baseline and Off-the-Shelf Approaches

To evaluate the efficacy of our generative diffusion-based approach, we first compare our pipeline against deterministic models and state-of-the-art, off-the-shelf depth and 3D reconstruction frameworks.

Deterministic Regression: As established, the orthographic projection of wireframes creates an inherently ill-posed problem, as a single 2D sketch can mathematically correspond to multiple valid 3D interpretations. When we attempted to train a standard deterministic regression model to map $x \rightarrow y$, the network consistently failed to resolve these ambiguities, instead converging to the mean of all possible modes.

Comparison with Off-the-Shelf Frameworks: We further trialed our wireframe inputs against leading zero-shot vision models, which share enough of our output contract for numeric and visual comparison.

Initially, we attempted to use an off-the-shelf depth estimator, namely the widely successful Depth Anything V2 [51]. While Depth Anything understands basic semantics of near and far regions, it is clearly unsuited to the unique visual domain of wireframe projections (Figure 5).

Additionally, we performed inference on Microsoft's Trellis [49] in an attempt to generate a 3D wireframe asset directly. Not only was the generation process too slow to support our intended "sketch-reconstruct-sketch" pipeline, but the results exhibited severe geometric distortions, underscoring a clear input unsuitability (Figure 5).

Comparison with Engineering Reconstruction Systems: For engineering-sketch systems with different output contracts, numeric comparison is undefined (Table 3); the comparison that is well-defined is what geometry each system demonstrates and what geometry its training distribution admits. The "Shape" column of Table 1 places representative

Table 4. **Quantitative benchmark results.** The upper panel reports the conditioning-encoder and scaling study: models trained on the v1 dataset at 256×256 and evaluated on the v1 test suite. The lower panel reports the expanded benchmark: models evaluated on the v2 dataset. Numbers are comparable within a panel, not across panels. The VAE floor rows report each autoencoder's round-trip reconstruction error on ground-truth depth maps, forming a hard lower bound on any diffusion model operating in that latent space; the 256 system's best-case error sits at its floor, while the new 512 VAE removes this ceiling and the fully expanded model converts it into a roughly $2\times$ reduction in best-case error.

	Model Details				Average Prediction			Best Prediction		
Model	**Params**	**Data**	**Res.**	**GPU Hrs**	**NMAE**	**AbsRel**	$\delta < 1.25$	**NMAE**	**AbsRel**	$\delta < 1.25$
v1 benchmark (10k shapes dataset)										
Baseline (average)	-	-	-	-	0.221 ± 0.001	0.262 ± 0.002	0.499 ± 0.004	0.221 ± 0.001	0.262 ± 0.002	0.499 ± 0.004
Depth Anything V2 [51]	98M	-	256	-	0.172 ± 0.002	0.208 ± 0.002	0.646 ± 0.006	0.172 ± 0.002	0.208 ± 0.002	0.646 ± 0.006
VAE encoder	63M	1M	256	112	0.101 ± 0.002	0.115 ± 0.002	0.863 ± 0.005	0.057 ± 0.002	0.065 ± 0.002	0.962 ± 0.003
ViT small scratch	84M	1M	256	128	0.100 ± 0.002	0.114 ± 0.002	0.868 ± 0.005	0.057 ± 0.002	0.065 ± 0.002	0.960 ± 0.003
DinoV2 small	84M	500K	256	128	0.072 ± 0.002	0.081 ± 0.002	0.927 ± 0.004	0.043 ± 0.001	0.047 ± 0.002	0.978 ± 0.002
DinoV2 small	84M	1M	256	128	0.069 ± 0.002	0.077 ± 0.002	0.941 ± 0.003	0.041 ± 0.001	0.047 ± 0.001	0.983 ± 0.002
DinoV3 small	84M	1M	256	128	0.070 ± 0.002	0.079 ± 0.002	0.937 ± 0.004	0.042 ± 0.001	0.048 ± 0.002	0.981 ± 0.002
DinoV2 base	227M	1M	256	336	0.059 ± 0.002	0.065 ± 0.002	0.956 ± 0.003	0.036 ± 0.001	0.039 ± 0.001	0.987 ± 0.001
DinoV3 base	227M	1M	256	336	0.066 ± 0.002	0.074 ± 0.002	0.940 ± 0.004	0.038 ± 0.001	0.043 ± 0.001	0.984 ± 0.002
DinoV2 base XL	**496M**	**1M**	**256**	**240**	$\mathbf{0.053 \pm 0.002}$	$\mathbf{0.058 \pm 0.002}$	$\mathbf{0.960 \pm 0.003}$	$\mathbf{0.031 \pm 0.001}$	$\mathbf{0.034 \pm 0.001}$	$\mathbf{0.989 \pm 0.001}$
DinoV3 base XL	496M	1M	256	240	0.067 ± 0.002	0.075 ± 0.002	0.937 ± 0.004	0.042 ± 0.001	0.046 ± 0.002	0.980 ± 0.002
Expanded v2 benchmark (90k shape dataset)										
DinoV2 base XL (no fine-tune)	496M	v1 (1M)	256	-	0.121 ± 0.001	0.123 ± 0.001	0.850 ± 0.002	0.091 ± 0.001	0.093 ± 0.001	0.920 ± 0.002
DinoV2 base XL (fine tuned)	496M	v2 (2.1M)	256	240	0.097 ± 0.002	0.095 ± 0.001	0.925 ± 0.001	0.072 ± 0.001	0.071 ± 0.001	0.965 ± 0.001
DinoV2 base XL high-res	**638M**	**v2 (2.1M)**	**512**	**310**	$\mathbf{0.070 \pm 0.001}$	$\mathbf{0.076 \pm 0.001}$	$\mathbf{0.932 \pm 0.001}$	$\mathbf{0.039 \pm 0.001}$	$\mathbf{0.040 \pm 0.001}$	$\mathbf{0.982 \pm 0.001}$
VAE floor (256 latent space)	-	-	256	-	0.069	0.069	0.979	0.069	0.069	0.979
VAE floor (512 latent space)	-	-	512	-	0.002	0.003	1.000	0.002	0.003	1.000

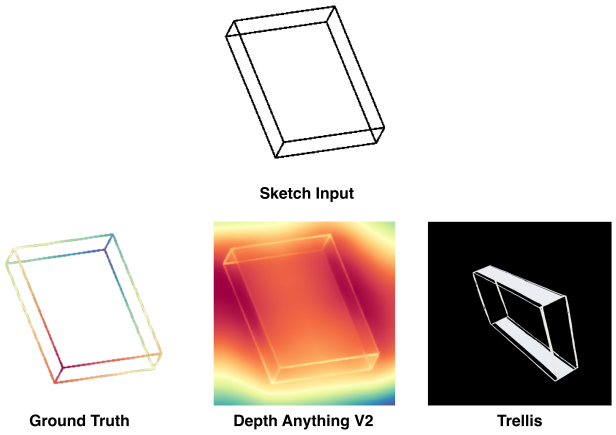


Figure 5. **Failures of off-the-shelf models.** For a simple rectangular prism, off-the-shelf models fail in both the depth estimation task (as predicted by Depth Anything V2) and direct 3D reconstruction (Trellis).

geometry from each closely related system's demonstrated domain alongside a reconstruction of ours. The prior domains concentrate on prismatic, low-curve-count geometry, consistent with their line/arc/circle program vocabularies and complexity-filtered training subsets (Table 2), whereas our method faithfully reconstructs a 60-stroke sketch in which 34 strokes are free-form B-spline curves from a single raster input. This is the practical consequence of the contract differences in Table 3: the geometry regime where engineering sketches are hardest is the regime our formulation is designed to cover.

7.2. Hyperparameter Analysis: Conditioning Encoders, Dataset Size, and Model Size

The choice of the conditioning encoder ($\mathcal{E}_{cond}$) dictates how effectively the U-Net denoiser can interpret the sparse, abstract nature of wireframe drawings. We tested three distinct encoder architectures: a Latent Encoder (VAE-KL), a ViT small trained from scratch, and a pre-trained DinoV2 backbone (Table 4).

Quantitative results (Table 4) indicate that the DinoV2 encoder achieved the best performance across all metrics. While the simpler, CNN-based VAE-KL encoder displayed an initially faster convergence speed than the DinoV2 encoder, DinoV2 eventually reached a lower final loss. This suggests that although modern foundation models are trained primarily on natural images, their learned representations translate to wireframe domains given sufficient fine-tuning. However, DinoV2's superior performance does not necessarily imply that a natural image prior is strictly better than a purely geometric prior, as architectural differences (ViT vs. CNN) play a significant role. Rather, the VAE-KL encoder's rapid initial convergence points to the ideal necessity of a wireframe-specific ViT backbone—one conditioned first on

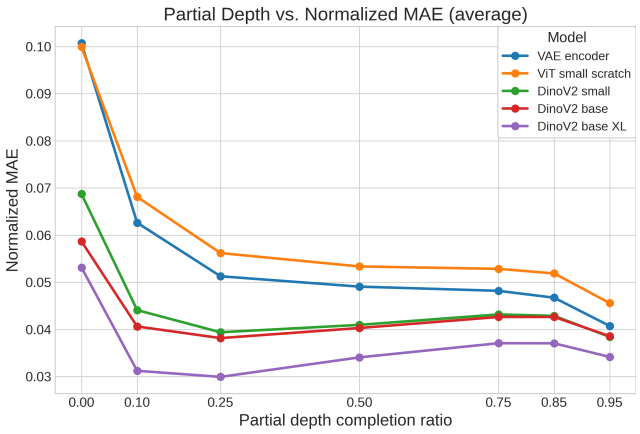


Figure 6. **Partial depth vs. Normalized MAE (average aggregation) for various encoders.** Overall, even minimal partial depth (10%–25%) provides a massive boost to reconstruction accuracy. However, increasing partial depth beyond this yields diminishing returns until reaching the final 95% threshold.

natural images and subsequently fine-tuned on geometric wireframes. Indeed, the fine-tuned DinoV2 encoder utilized in this work fills precisely this gap, representing a preliminary step toward a wireframe-specific vision foundation model.

Ultimately, while encoders trained on natural images provide a suitable foundation, they are limited in the long run. Even as we ran a benchmark with the newer DinoV3, trained on nearly 12× the amount of data compared to the V2, it resulted in virtually no improvement in the small and base models and a significant deterioration in the base XL. Thus, to improve performance in wireframe specific tasks, it is not enough to simply build a better natural image-based model; it is instead required to build better wireframe specific foundation models.

Furthermore and predictably, increasing model size improved performance while decreasing dataset size harmed performance. This trend is consistent with the scaling behavior commonly observed in large generative models [1]. Whereas approaches dependent on stroke primitives must be manually adapted and retrained for new curve topologies, our approach offers a straightforward path for continuous improvement via increased computational capacity, either in parameter size or dataset size.

7.3. Evaluation of Partial Depth and Sequential Sketching

A key contribution of our method is the ability to intuitively reconstruct from partial depth, enabling a "sketch-reconstruct-sketch" workflow. To validate this, we evaluated the v1 models' performance while varying the partial depth threshold $k \sim U[10\%, 90\%]$ using our graph-based BFS masking strategy.

As the percentage of provided partial depth (k) increases, the NMAE predictably decreases, with every model approaching zero error as the provided partial depth nears completion (Figure 6). However, this decrease is non-linear. Providing the depth of just 10% of the sketch acts as a geometric "anchor" that sufficiently boosts performance, while providing 25% of the sketch achieves similar performance to providing 75%.

However, between the 25% and 85% thresholds, there exists an interesting plateau and even slight degradation in performance. This plateau suggests that once the global structural details are anchored (accounted for in the first 10% to 25%), remaining errors are likely concentrated on smaller structural details that are objectively more difficult to predict, such as a disjointed chunk of the wireframe's edges.

These results show that partial depth cues provide substantial improvement during iterative reconstruction, validating the success of the "sketch-reconstruct-sketch" workflow. Supporting iterative sketching not only allows the user to comfortably tackle complex objects in stages, but the iterative geometric anchoring ensures the model autoregressively avoids compounding structural mistakes.

The partial depth curves also illuminate an interesting finding. The first jump in model size (from 84M to 227M) was marked primarily by an increase in zero-shot performance (0% completion ratio), while the second jump in model size (from 227M to 496M) was marked primarily by increases in non-zero-shot tasks where there was existing partial depth.

7.4. Robustness to Complexity

To test the robustness of our pipeline against sketches of varying difficulty, we segmented the v1 test set by shape complexity and APR (Figure 7).

There exists a positive relationship between predictive error and both shape complexity ($R \approx 0.323$–0.337) and APR ($R \approx 0.362$–0.388) (Figure 7a), implying the model is understandably prone to both parameters. However, while shape complexity is constant and unchangeable, APR can be decreased and mediated by strategies such as drawing from better viewpoints, increasing input resolution, and lowering stroke thickness.

The APR and the complexity of the shape are naturally correlated; thus, the question becomes whether reducing the APR can lower the error for shapes of various complexities. If so, the strategies presented provide a straightforward way to improve the model. Thus, in Figure 7b, we show that regardless of the low / medium / high complexity shapes, the APR and the error continue to be correlated; therefore, by reducing the APR even for the high complexity shapes, we can effectively reduce the error.

Ultimately, while the complexity of a shape is a main driver for error, APR makes up a significant portion of why

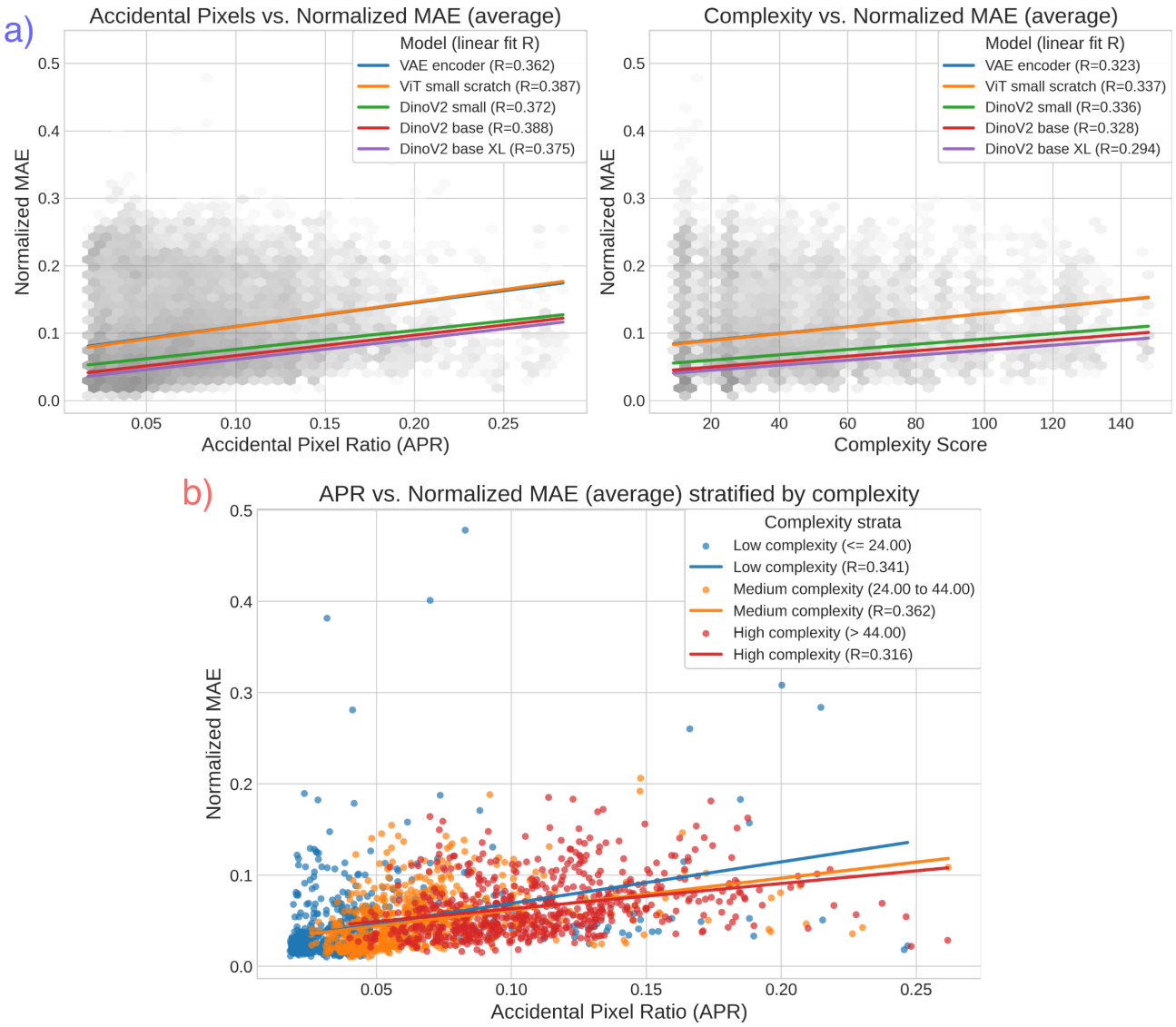


Figure 7. **APR and complexity analysis.** a) APR & shape complexity score vs. normalized MAE for various encoders. Each plot displays density bins pooled across predictions of all models with x-axis outliers removed and b) APR vs. normalized MAE binned by shape complexity for the best DinoV2 model. While APR and complexity are both correlated with increasing error, reducing APR provides a simple path to improvement even for complex shapes.

complexity is such a significant cause for high error.

7.5. Scaling Resolution and Data

The preceding analyses identified accidental pixel density as one structural bottleneck of the 256×256 system. Measuring the autoencoder's round-trip reconstruction error on ground-truth depth maps reveals a second: the 256 system's best-case error sits essentially on its VAE reconstruction floor, so no amount of further diffusion training could improve it. This motivates the full 512-resolution system of Section 5: the expanded curve-rich dataset, a retrained VAE with an order-of-magnitude lower floor, and a U-Net retrained from scratch in the new latent space.

The lower panel of Table 4 compares three systems on the expanded v2 benchmark under identical protocol, comparing the original DinoV2 base XL trained on v1, the same architecture fine-tuned on the expanded v2 data at 256×256, and our 512-resolution model.

Two observations follow. First, expanding the training data alone helps, but the 256 system's best-case accuracy converges to its VAE reconstruction floor. Second, replacing the latent space and doubling resolution converts that headroom into an approximately $2\times$ reduction in best-of-five error across all metrics, with the largest relative gains ex-

actly where the APR analysis predicted: dense, curve-heavy structures that were previously unresolvable at 256×256.

7.6. Qualitative Results

We also evaluate qualitative reconstruction behavior by visualizing predicted depth maps, reconstructed point clouds, and vectorized wireframes (Figure 8). The wireframe is constructed by assigning depth-lifted pixels to individual strokes, filtering pixels whose depth is contaminated by occluding strokes, resolving a shared depth value for each junction of the stroke graph, and fitting a parametric primitive (line, circle/arc, or B-spline) to each stroke's surviving 3D support with endpoints pinned to the resolved junctions.

Overall, the model produces high-quality reconstructions across a range of engineering wireframes, including heavily curved and organic geometry. As deduced in the previous sections, residual errors remain concentrated where accidental pixels are dense: the binary input sketch mask cannot differentiate between overlapping 2D strokes, so crossing-heavy regions correspond to the noisiest portions of the reconstructed point clouds (bottom-right example in Figure 8). The migration from 256×256 to 512×512 conditioning with thinner strokes substantially reduced this failure mode, confirming the resolution hypothesis raised in the v1 analyses of this work, but does not eliminate it entirely for the densest structures.

8. Limitations and Future Work

Although our framework presents a highly flexible alternative for the 2D-to-3D reconstruction of wireframes, it also highlights specific challenges inherent to depth-based representations.

Sketch input noise. Human freehand sketches are inherently noisy and highly dependent on an individual's drawing skill. Because our current model is trained exclusively on clean, rendered wireframe projections, it can struggle to generalize to highly irregular or inaccurate freehand input, manifesting in wiggly lines and geometry distortions. To mitigate this, our current interface provides drawing assisting features, such as stroke normalization, endpoint snapping, and an isometric grid, to guide user measurements. Our current formulation preserves the input stroke support by construction, including any user-introduced noise. To truly develop an end-to-end framework capable of capturing a user's underlying design *intent*, future iterations of the diffusion model must learn to map noisy inputs to idealized clean geometries. Training such a model requires perturbing clean wireframes with simulated noise (e.g., vertex jitter and edge wobble). Preliminary experiments suggest a complex spectrum of "noise," ranging from high-frequency stroke perturbations (wobbly lines) to low-frequency structural distortions (e.g., skewing a rectangular face into a parallelogram). Consequently, future models may require explicit noise-conditioning scalars and must learn stronger geometric priors to regularize internal topologies toward what makes a geometrically regular engineering wireframe.

Model size and resolution. Upon scaling our architecture to a larger parameter count, we observed a distinct improvement in our accuracy metrics, and the migration from 256×256 to 512×512 (Section 7.5) confirmed that resolution and latent-space fidelity were the binding bottlenecks, roughly halving best-case error. This indicates that our framework benefits from the scaling laws commonly observed in large foundation models, offering a straightforward path to further improvement via increased computational capacity [1].

Post-processing and mesh conversion. Our vectorization stage fits parametric primitives (lines, circular arcs, and true B-splines with endpoint-constrained least-squares control points) to the depth-lifted strokes, with occlusion-aware pixel attribution and shared junction depths. Its main remaining weakness is inherited from the depth stage: where overlapping strokes contaminate depth, primitive fits degrade gracefully but visibly. Furthermore, once a mathematically precise wireframe is established, inferring which planar cycles constitute solid faces is required to generate a final, watertight mesh. In this regard, our generative depth approach could be highly complementary when paired with topological sequence models like Neural Face Identification [46], or even more traditional face identification techniques like in Shpitalni & Lipson [39]. However, aggressive post-processing introduces a representation tradeoff: enforcing rigid parametric constraints diminishes the primary advantage of our generative approach—its ability to model partial, open, and non-standard wireframe structures without predefined vocabularies. Navigating this balance will ultimately depend on the specific downstream needs of the end user.

Conditioning input. To maintain a fully unconstrained pipeline, we deliberately limited our geometric conditioning inputs to binary input sketch masks and partial depth. However, providing explicit stroke-level instance segmentation (i.e., mapping specific pixels to individual strokes and providing the order of strokes drawn) could resolve significant geometric ambiguities. This would alleviate granularity bottlenecks, allowing the network to easily disentangle dense pixel clusters into distinct edges. While requiring users to draw each edge with a single, continuous stroke aligns well with natural human drawing behavior, it reintroduces a subtle operational constraint, lightly compromising the vision of a purely unconstrained predictive pipeline.

9. Conclusions

In this paper, we presented a projection-faithful approach to reconstructing 3D geometry from engineering line drawings. Rather than treating sketch interpretation as semantic ob-

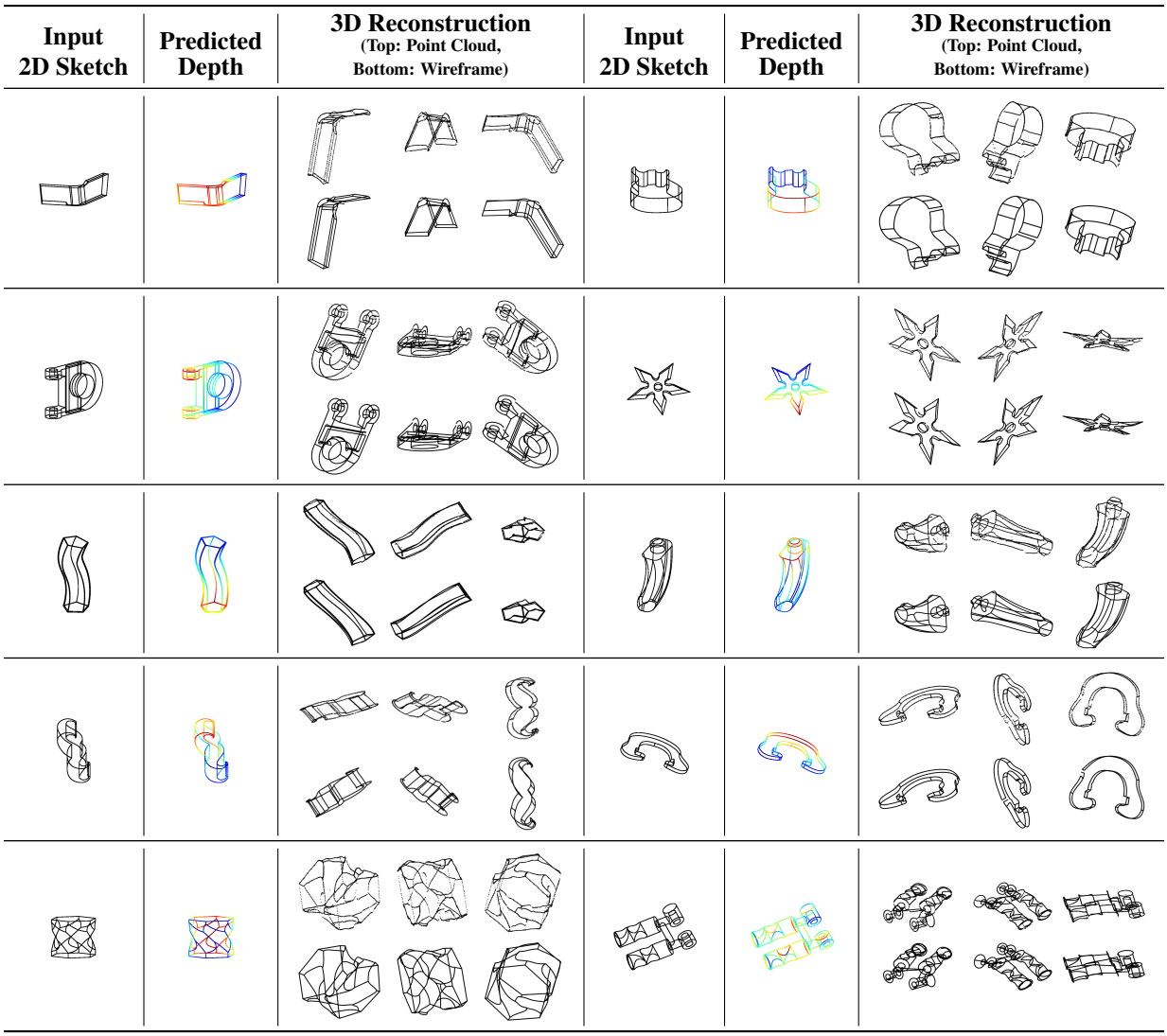


Figure 8. **Qualitative results of our** 512×512 **model.** Each block shows the input sketch, the predicted disparity over the drawn strokes, and the 3D reconstruction rendered from three fixed viewpoints at a single shared scale: the raw point cloud unprojected from the predicted depth (top) and the vectorized wireframe fitted with parametric line, circle, and B-spline primitives (bottom). The model reconstructs curved, organic, and thin geometry faithfully; residual errors concentrate in regions dense with overlapping strokes, as in the bottom-right example.

ject completion or CAD program induction, we formulated the task as conditional depth estimation on visible sketch support, allowing a sparse 2D drawing to be lifted directly into an embedded 3D wireframe. This narrower formulation matches an important early-stage engineering need: preserving what the user drew while inferring only the missing depth needed for 3D reasoning.

Methodologically, the paper shows that latent diffusion with spatial sketch conditioning is a strong fit for this problem. It provides a generative prior for an inherently multimodal reconstruction task, benefits from large-scale rendered engineering data, and naturally supports partial-depth conditioning for iterative design. Empirically, the method becomes substantially more accurate once even limited partial depth is available, validating the sketch-reconstruct-sketch workflow, and scales predictably: expanding to a curve-rich 90,000-shape corpus, retraining the latent autoencoder, and doubling resolution to 512×512 roughly halves reconstruction error, reaching 3.9% best-of-five normalized depth error on 4,528 unseen shapes.

Ultimately, this work provides a scalable alternative to traditional parametric modeling by enabling the reconstruction of partial, open, and non-standard wireframe structures without the "vocabulary bottleneck" of standard CAD primitives. By reducing the technical barrier between a 2D stroke and its 3D realization, this work moves toward a more intuitive

interface where users can fluidly and iteratively "draw in 3D."

10. Acknowledgements

This work was supported by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (DynamicsAI.org) under grant 2112085. Code is available at https://eltonc01.github.io/sketch-depth-diffusion

References

[1] X. Bi et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[2] Minglin Chen, Weihao Yuan, Yukun Wang, Zhe Sheng, Yisheng He, Zilong Dong, Liefeng Bo, and Yulan Guo. Sketch2nerf: multi-view sketch-guided text-to-3d generation. *arXiv preprint arXiv:2401.14257*, 2024.

[3] Yang Chen, Yingwei Pan, Yehao Li, Ting Yao, and Tao Mei. Control3d: Towards controllable text-to-3d generation. *arXiv preprint arXiv:2311.05461*, 2023.

[4] Maxwell B. Clowes. On seeing things. In *Artificial Intelligence*, pages 79–116, 1971.

[5] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A. Efros, and Adrien Bousseau. 3d sketching using multi-view deep volumetric prediction. In *Proceedings of the ACM SIGGRAPH Symposium on Sketch-Based Interfaces and Modeling*, 2017.

[6] Zhi Deng, Yang Liu, Hao Pan, Wassim Jabi, Juyong Zhang, and Bailin Deng. Sketch2pq: Freeform planar quadrilateral mesh design via a single sketch. *arXiv preprint arXiv:2201.09367*, 2022.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, and J. Uszkoreit. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[9] K. Eissen and R. Steur. *Sketching: Drawing techniques for product designers*. BIS Publishers, 2008.

[10] Chenjian Gao, Qian Yu, Lu Sheng, Yi-Zhe Song, and Dong Xu. Sketchsampler: Sketch-based 3d reconstruction via view-dependent depth sampling. In *European conference on computer vision*, pages 464–479. Springer, 2022.

[11] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. *arXiv preprint arXiv:2104.00482*, 2021.

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] D. A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[14] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 409–416, 1999.

[15] Ahmet Serdar Karadeniz, Dimitrios Mallis, Nesryne Mejri, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Picasso: A feed-forward framework for parametric inference of cad sketches via rendering self-supervision. *arXiv preprint arXiv:2407.13394*, 2024.

[16] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics*, 25(3):589–598, 2006.

[17] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9492–9502, 2024.

[18] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

[19] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, A. Somov, D. Zorin, and D. Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

[20] S. Koley, T. K. Dutta, A. Sain, P. N. Chowdhury, A. K. Bhunia, and Y. Z. Song. Sketchfusion: Learning universal sketch features through fusing foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2556–2567, 2025.

[21] C. Li, H. Pan, A. Bousseau, and N. J. Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[22] C. Li, H. Pan, A. Bousseau, and N. J. Mitra. Free2cad: Parsing freehand drawings into cad commands. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.

[23] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. *arXiv preprint arXiv:2303.10613*, 2023.

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

[25] H. Lipson and M. Shpitalni. Correlation-based reconstruction of a 3d object from a single freehand sketch. In *ACM SIGGRAPH 2007 Courses*, pages 44–es, 2007.

[26] Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. Sketchdream: Sketch-based text-to-3d generation and editing. *arXiv preprint arXiv:2405.06461*, 2024.

[27] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[28] Mark Masry and Hod Lipson. A sketch-based interface for iterative design and analysis of 3d objects. In *ACM SIGGRAPH 2007 courses*, pages 31–es. 2007.

[29] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan. Structure-aware shape processing. *Eurographics State of the Art Reports*, 32(2):1–21, 2013.

[30] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: Designing freeform surfaces with 3d curves. *ACM Transactions on Graphics*, 26(3):41, 2007.

[31] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[32] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, and P. Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, S. Chintala, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[34] Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A. Yeh, and Rana Hanocka. Geocode: Interpretable shape programs. *arXiv preprint arXiv:2212.11715*, 2022.

[35] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2020.

[36] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[37] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[38] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 234–241. Springer, 2015.

[39] M. Shpitalni and H. Lipson. Identification of faces in a 2d line drawing projection of a wireframe object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1000–1012, 1996.

[40] O. Siméoni et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[41] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[42] K. Sugihara. *Machine Interpretation of Line Drawings*. MIT Press, 1986.

[43] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, pages 329–346, 1963.

[44] Alberto Tono, Heyaojing Huang, Ashwin Agrawal, and Martin Fischer. Vitruvio: 3d building meshes via single perspective sketches. *arXiv preprint arXiv:2210.13634*, 2022.

[45] Alberto Tono, Jiajun Wu, Gordon Wetzstein, Iro Armeni, Hariharan Subramonyam, James Landay, and Martin Fischer. Deep sketch-based 3d modeling: A survey. *arXiv preprint arXiv:2603.03287*, 2026.

[46] X. Wang et al. Neural face identification in a 2d wireframe projection of a manifold object. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[47] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[48] R. Wu, C. Xiao, and C. Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 6772–6782, 2021.

[49] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, et al. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 21469–21480, 2025.

[50] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024.

[51] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024.

[52] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 163–170, 1996.

[53] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[54] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. Sketch2model: View-aware 3d modeling from single free-hand sketches. *arXiv preprint arXiv:2105.06663*, 2021.

[55] Wangguandong Zheng, Haifeng Xia, Rui Chen, Ming Shao, Siyu Xia, and Zhengming Ding. Sketch3d: Style-consistent guidance for sketch-to-3d generation. *arXiv preprint arXiv:2404.01843*, 2024.

[56] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *arXiv preprint arXiv:2305.04461*, 2023.

[57] Yue Zhong, Yonggang Qi, Yulia Gryaditskaya, Honggang Zhang, and Yi-Zhe Song. Towards practical sketch-based 3d shape generation: The role of professional sketches. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(9):3518–3528, 2020.