PARASKETCHFM: JOINT CONTINUOUS–DISCRETE FLOW MATCHING FOR PARAMETRIC CAD SKETCH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite substantial progress in parametric CAD sketch generation, existing generators still exhibit primitive-type distribution discrepancies on SketchGraphs, leaving room to improve generation quality and distributional fidelity. We therefore introduce PARASKETCHFM, a joint discrete–continuous flow-matching model with geometric regularization. The model couples absorbing categorical paths with continuous geometric flows over unordered primitive sets and incorporates a one-sided anchor-distance regularizer, addressing two key challenges: matching primitive-type distributions and mitigating geometric contraction during joint generation. Under a shared evaluation protocol on SketchGraphs, PARASKETCHFM achieves an FID of 5.22, precision of 0.431, and recall of 0.490, with a maximum per-type slot-frequency deviation from the test set of 0.31 percentage points. These results establish a new performance benchmark under this protocol.

1 INTRODUCTION

Parametric CAD sketches describe geometry through editable primitives such as lines, circles, arcs, and points (Figure 1). Generating these primitives can support design exploration and subsequent constraint modeling. Existing methods have made substantial progress in producing plausible sketches, yet our evaluation on SketchGraphs reveals differences between generated and reference primitive-type frequencies. We hypothesize that imperfect recovery of primitive composition contributes to the remaining gap in generation quality. This motivates improving generation quality and distributional fidelity together: we use FID as the primary quality metric, complemented by precision, recall, and primitive-type statistics. We generate primitives and construction labels, and do not model geometric constraints.

Autoregressive generators such as Vitruvion and UniSketch predict each token from the preceding sequence (Seff et al., 2022; Lin et al., 2026). Autoregressive factorization is expressive enough to represent the target joint distribution, but finite models trained on imbalanced data depart from the ideal conditionals, and conditioning on generated prefixes compounds these errors. In our reproductions, Vitruvion and the UniSketch port allocate 5.65 and 6.65 percentage points more of all slots to lines than the test set, respectively, while circles, arcs, and points are underrepresented (Section 4.2). These observations motivate exploring joint generation without a prescribed primitive order.

SketchDNN makes an important step in this direction through a permutation-equivariant set model that jointly denoises geometry and Gaussian–Softmax representations of categorical attributes (Chereddy & Femiani, 2025). This formulation avoids dependence on a prescribed primitive order and achieves broader coverage than the autoregressive reproductions in our evaluation. It does not exhibit the same overrepresentation of lines; instead, circles, arcs, and points occur more frequently than in the reference data. Stochastic denoising explores diverse configurations, but in a learned, finite-step sampler it also interacts with prediction errors, and the long reverse chain in our reproduction incurs substantial sampling cost. These observations motivate examining generation paths that reduce repeated noise injection.

Flow matching learns transport along prescribed probability paths and enables generation by ODE integration (Lipman et al., 2023). For geometry, it replaces stochastic reverse updates with a learned vector field, eliminating fresh Gaussian noise injection during integration. We adopt this formulation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

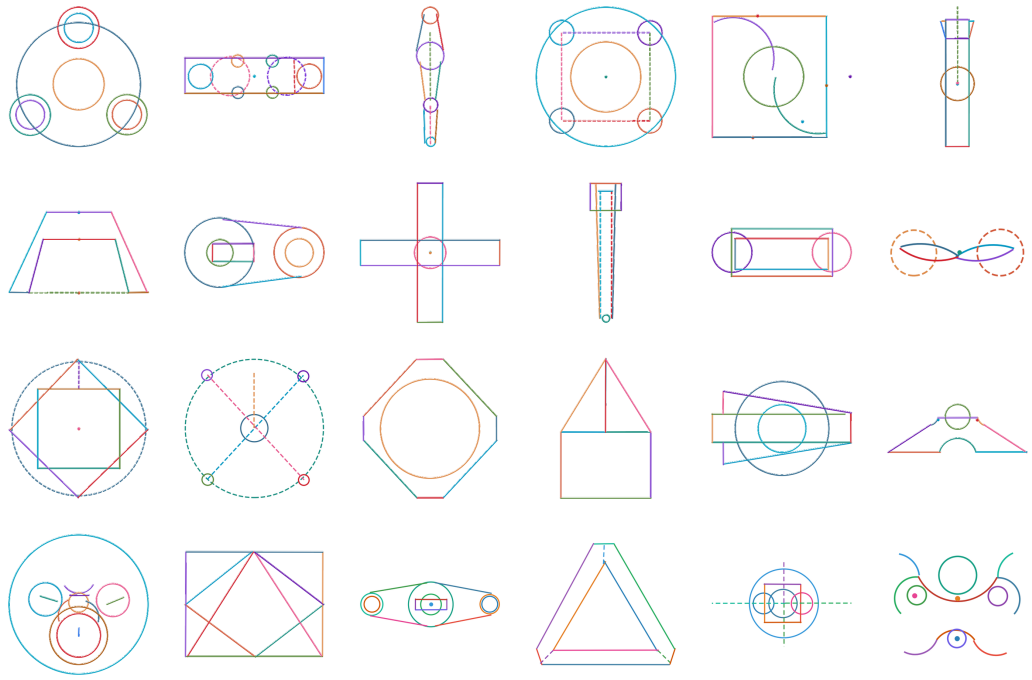

Figure 1: Unconditional sketch generation. Colors identify primitives and dashed curves denote construction geometry.

while keeping the set-based representation. The central modeling challenge is how to coordinate discrete attributes and continuous geometry within the same generation process: categories must resolve into valid primitive types, while the corresponding geometric parameters must form coherent sketches.

We compare three path allocations under a common backbone and training configuration. Continuous-all (CFM) follows the continuous categorical-representation idea of SketchDNN's Gaussian–Softmax formulation: we encode labels as smoothed, centered logits and model both attributes and geometry with continuous flows. Discrete-all (DFM) quantizes geometry into bins and uses the discrete flow-matching framework of Gat et al. (2024) to generate all fields categorically. Joint flow matching (JFM) preserves the discrete nature of attributes and the continuous parameterization of geometry, coupling absorbing categorical paths with a geometric ODE through a shared predictor. Following SketchDNN, we represent each slot using separate type-specific geometry blocks; type probabilities weight these blocks during joint updates.

The comparisons reveal different consequences of these choices. In our discrete-all implementation, quantization and categorical generation exacerbate the type-frequency discrepancy: 99.78% of slots are assigned to lines and recall falls to 0.006. The continuous-all formulation instead improves both primitive composition and generation quality relative to our SketchDNN reproduction, with a maximum per-type frequency deviation of 0.16 percentage points and an FID of 8.30 rather than 12.05. These results support using flow matching for this representation, but matching type frequencies alone does not fully determine generation quality. The joint formulation offers a further improvement when paired with geometric regularization.

Without the pair regularizer, JFM has an FID of 23.99, compared with 8.18 for CFM without the same term. Examining generated samples reveals a subset of sketches with nearly point-like extents, suggesting geometric contraction as a contributor to this quality gap. We therefore introduce a one-sided anchor-distance regularizer that penalizes predicted distances that are too small relative to the target geometry. At the selected coefficient, JFM's FID improves to 5.22, and the fraction of near-point sketches falls from 2.37% to 0.49%. Applying the same regularizer to CFM provides no comparable FID improvement over the tested range: its FID remains between 8.18 and 8.30. The

resulting regularized joint model, PARASKETCHFM, combines close recovery of primitive-type frequencies with improved sketch geometry, achieving the best FID among the evaluated methods (Section 4.3).

Our contributions are:

- We identify primitive-type frequency discrepancies in existing sketch generators and develop a method that substantially reduces them, matching the SketchGraphs reference frequencies to within 0.31 percentage points per type.
- We develop a flow-matching framework for parametric CAD sketch generation and formulate a regularized joint discrete–continuous model. Comparisons with fully continuous and fully discrete alternatives show that the regularized joint formulation provides the best generation quality among the evaluated configurations.
- We establish a new performance benchmark on SketchGraphs under a shared evaluation protocol, achieving FID 5.22, precision 0.431, and recall 0.490.

2 RELATED WORK

2.1 PARAMETRIC CAD SKETCH GENERATION

SketchGraphs represents human-authored Onshape sketches as graphs of geometric primitives and constraints (Seff et al., 2020). Early sequence-based generators serialize drawing commands (Willis et al., 2021a) or primitive and constraint tokens (Para et al., 2021). Related CAD generators instead model command sequences or boundary representations (Wu et al., 2021; Xu et al., 2022; Ganin et al., 2021; Jayaraman et al., 2023), supported by datasets of human CAD construction sequences (Willis et al., 2021b). These works establish effective generative models for structured CAD data.

Vitruvion is a representative autoregressive model for parametric sketches: it first generates primitives with their initial geometry and then generates constraints that reference those primitives, yielding sketches that can be solved and edited in standard CAD systems (Seff et al., 2022). UniSketch extends this paradigm to seven primitive and 23 constraint types (Lin et al., 2026), while CADSketcher supports bidirectional query-driven completion (Liu et al., 2026).

SketchDNN is our closest comparison. It replaces a prescribed primitive order with up to 16 unordered slots, uses type-specific geometry blocks, and jointly denoises continuous geometry and Gaussian–Softmax representations of categorical attributes (Chereddy & Femiani, 2025). We inherit its sketch representation and permutation-equivariant backbone, and study how probability-path allocation and geometric regularization affect primitive composition and generation quality.

2.2 PROBABILITY PATHS FOR MIXED STRUCTURED DATA

Gaussian diffusion, score-based SDEs, and Flow Matching model continuous data (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023), while D3PM, masked diffusion, and Discrete Flow Matching provide categorical counterparts (Austin et al., 2021; Shi et al., 2024; Sahoo et al., 2024; Lou et al., 2024; Gat et al., 2024). Related continuous formulations include rectified flow and stochastic interpolants (Liu et al., 2023; Albergo et al., 2025).

Some methods place all variables in one state space. SketchDNN, FlowMol, and LayoutFlow use continuous categorical representations (Chereddy & Femiani, 2025; Dunn & Koes, 2024a; Guerreiro et al., 2024), as do analog-bit encodings of discrete data (Chen et al., 2023), whereas LayoutDM quantizes continuous layout coordinates (Inoue et al., 2023). Our CFM baseline retains SketchDNN's continuous categorical representation strategy while replacing its diffusion paths with affine flow-matching paths; our DFM baseline adapts Gat et al. (2024) to parametric CAD by quantizing active geometry and evolving every field categorically.

Mixed models instead retain continuous and categorical variables. A staged alternative generates one modality and predicts the other conditionally; DLT finds joint class–box diffusion preferable to its sequential variants (Levi et al., 2023). DLT, MiDi, and equivariant molecule diffusion jointly evolve continuous geometry and discrete attributes (Levi et al., 2023; Vignac et al., 2023; Hoogeboom et al., 2022); Multiflow and FlowMol-CTMC combine discrete flows with continuous structural or

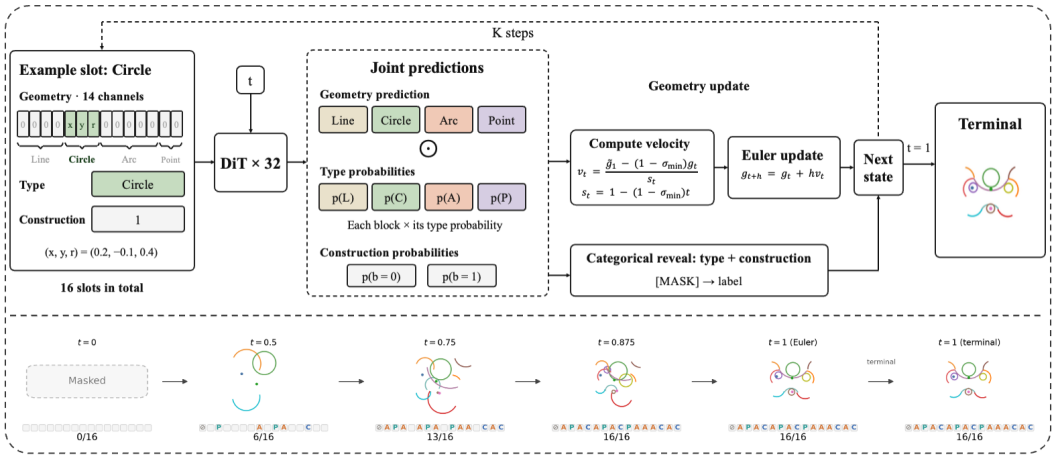

Figure 2: Joint generation of primitive types and geometry. **Top:** one sampling step. A shared DiT reads all 16 slots together with the time t and predicts a clean geometry endpoint, type probabilities, and construction probabilities. Each geometry block is weighted by its type probability before the velocity is formed and integrated with an Euler update, while masked categorical fields are revealed in the same step. **Bottom:** an illustrative trajectory with the number of revealed slots below each state. Geometry evolves both before and after type revelation, and one terminal call removes the residual endpoint noise. Slot labels denote revealed types (∅=NONE, L=line, C=circle, A=arc, P=point).

coordinate flows for proteins and molecules (Campbell et al., 2024; Dunn & Koes, 2024b). These studies establish the effectiveness of joint continuous–categorical generation.

To our knowledge, PARASKETCHFM is the first joint discrete–continuous flow-matching method for parametric CAD sketches. We adapt mixed flows to SketchDNN's type-indexed geometry and, under one representation and backbone, compare continuous-all, discrete-all, and joint paths. In this comparison the joint path becomes the best-performing configuration only once a one-sided anchor-distance regularizer removes its near-point contraction. This controlled comparison and regularized CAD adaptation distinguish our work from prior mixed-flow studies. Freehand and vector-graphics generators use other representations and objectives (Ha & Eck, 2018; Carlier et al., 2020; Zhou et al., 2026).

3 PARASKETCHFM

PARASKETCHFM uses a shared permutation-equivariant DiT to predict geometry, primitive type, and construction status from a mixed continuous–categorical state. Figure 2 summarizes the model and joint sampler.

3.1 PROBLEM FORMULATION AND SKETCH REPRESENTATION

A sketch is an unordered collection of $L = 16$ primitive slots, modeled as a permutation-invariant set (Zaheer et al., 2017). Slot i contains a binary construction flag b_i, a primitive type c_i, and a geometry vector $\mathbf{g}_i \in \mathbb{R}^{14}$. The type set is $\{\text{LINE}, \text{CIRCLE}, \text{ARC}, \text{POINT}, \text{NONE}\}$, where NONE fills unused slots and therefore encodes sketch cardinality.

The geometry vector concatenates one parameter block per real primitive type, with widths $4, 3, 5, 2$ for line, circle, arc, and point, respectively. Only the block selected by c_i is active; all other coordinates are zero in clean data. Each categorical field receives one additional `[MASK]` state, which denotes an unresolved value, whereas NONE denotes a resolved empty slot. No slot positional encoding is used.

3.2 CONTINUOUS AND CATEGORICAL PROBABILITY PATHS

Let $\mathbf{G}_1 \in \mathbb{R}^{L\times 14}$ collect the clean geometry vectors and let $\mathbf{E} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a Gaussian source of the same shape. Geometry follows the affine Gaussian conditional path of Lipman et al. (2023):

$$\mathbf{G}_t = s_t\mathbf{E} + t\mathbf{G}_1, \qquad s_t = 1 - (1 - \sigma_{\min})t. \tag{1}$$

For a categorical field $r \in \{\text{type}, \text{constr}\}$, let $z_{t,i}^{(r)}$ denote the state of slot i at time t, let $z_{1,i}^{(r)}$ be its clean value, and let m_r be the field-specific mask. The field follows the absorbing path

$$q_t\left(z_{t,i}^{(r)} \mid z_{1,i}^{(r)}\right) = (1-t)\delta_{m_r}\left(z_{t,i}^{(r)}\right) + t\delta_{z_{1,i}^{(r)}}\left(z_{t,i}^{(r)}\right), \tag{2}$$

where δ_a is a point mass at state a. Conditional on the clean sketch and t, masks are sampled independently across slots and categorical fields. Together, $\mathbf{G}_t$ and the two categorical fields form the mixed state $\mathbf{X}_t$.

As in standard Gaussian conditional paths, a positive $\sigma_{\min}$ keeps the endpoint-to-velocity conversion bounded near $t = 1$. It leaves residual noise $\sigma_{\min}\mathbf{E}$ at $t = 1$, which the terminal denoising step of Section 3.5 removes.

A permutation-equivariant DiT (Peebles & Xie, 2023) consumes $(\mathbf{X}_t, t)$ and predicts

$$F_\theta(\mathbf{X}_t, t) = \left(\widehat{\mathbf{G}}_1, \boldsymbol{\ell}^{\text{type}}, \boldsymbol{\ell}^{\text{constr}}\right). \tag{3}$$

Here $\widehat{\mathbf{G}}_1$ is the predicted clean geometry, and $\boldsymbol{\ell}^{\text{type}}$ and $\boldsymbol{\ell}^{\text{constr}}$ are logits for the clean categorical values. `[MASK]` is never a prediction target.

3.3 TRAINING OBJECTIVE AND GEOMETRIC REGULARIZATION

We minimize

$$\mathcal{L} = \mathcal{L}_{\text{geom}} + \mathcal{L}_{\text{type}} + \mathcal{L}_{\text{constr}} + \eta\mathcal{L}_{\text{pair}}. \tag{4}$$

Let $M(c_i) \in \{0,1\}^{14}$ select the geometry block associated with the clean type c_i, and let $D(\mathbf{C}) = \sum_i \|M(c_i)\|_1$ be the number of active geometric coordinates in a sketch. The geometric endpoint-regression loss is

$$\mathcal{L}_{\text{geom}} = \mathbb{E}\left[\lambda(t)\frac{\sum_i \|M(c_i)\odot(\widehat{\mathbf{g}}_{1,i} - \mathbf{g}_{1,i})\|_2^2}{D(\mathbf{C})}\right], \qquad \lambda(t) = \bar{\lambda}_g\frac{\sigma_{\min}}{s_t^2}. \tag{5}$$

The expectation is over the data, time, and path randomness. The factor s_t^{-2} converts endpoint error to velocity error, and $\sigma_{\min}$ normalizes its mean under uniform time since $\int_0^1 \sigma_{\min}s_t^{-2}\,\mathrm{d}t = 1$. The type and construction losses are slot-averaged cross-entropies on the clean categorical values, computed from both masked and revealed inputs and weighted uniformly over time.

To discourage near-point contraction in predicted sketches, we add a one-sided anchor-distance regularizer. Ground-truth types select corresponding target and predicted anchors $\mathbf{a}_r$ and $\widehat{\mathbf{a}}_r$. For a sampled set $\mathcal{P}$ of valid anchor pairs,

$$\mathcal{L}_{\text{pair}} = \frac{1}{|\mathcal{P}|}\sum_{(r,s)\in\mathcal{P}}\left[\log d_\epsilon(\mathbf{a}_r, \mathbf{a}_s) - \log d_\epsilon(\widehat{\mathbf{a}}_r, \widehat{\mathbf{a}}_s) - \mu\right]_+^2, \tag{6}$$

where d_ϵ is a floored Euclidean distance, $[u]_+ = \max(u, 0)$, and $\mu = 0.05$. The loss penalizes predicted distances that are too small without penalizing larger ones. Appendix B.3 gives the anchor definitions, filtering rules, and remaining coefficients.

3.4 INFERENCE-TIME TYPE-PROBABILITY GATING

The training objective predicts raw geometry endpoints and categorical logits; type-probability gating is applied only during sampling. Let $x = (\mathbf{X}_t, t)$, let τ be a real primitive type, and let $\mathbf{g}_{1,i}^\tau$ be its geometry block in slot i. Because every other type stores zero in that block, the clean-data distribution satisfies

$$\mathbb{E}[\mathbf{g}_{1,i}^\tau \mid x] = \Pr(c_{1,i} = \tau \mid x)\,\mathbb{E}[\mathbf{g}_{1,i}^\tau \mid x, c_{1,i} = \tau], \tag{7}$$

where $\Pr(c_{1,i} = \tau \mid x)$ is the conditional probability that the clean type of slot i is τ. This decomposition motivates using the predicted type probabilities to weight the corresponding geometry blocks during inference.

Let $\mathbf{p}_i^{\text{type}} = \text{softmax}(\boldsymbol{\ell}_i^{\text{type}})$. We define the gated endpoint

$$\widetilde{\mathbf{g}}_{1,i} = W(\mathbf{p}_i^{\text{type}}) \odot \widehat{\mathbf{g}}_{1,i}, \qquad W(\mathbf{p}) = [p_{\text{line}}\mathbf{1}_4, p_{\text{circle}}\mathbf{1}_3, p_{\text{arc}}\mathbf{1}_5, p_{\text{point}}\mathbf{1}_2], \tag{8}$$

where $\mathbf{1}_d$ is a length-d all-ones vector and $\odot$ denotes elementwise multiplication. The NONE probability has no geometry block. We apply this gate before every geometric Euler update and again in the terminal denoising call.

3.5 JOINT SAMPLING

Generation starts from Gaussian geometry $\mathbf{G}_0 = \mathbf{E}$ and fully masked categorical fields. With K steps, let $t_k = k/K$ and $h = 1/K$. At step k, one network call produces a raw geometry endpoint and both categorical distributions; Equation 8 first converts the raw endpoint to $\widetilde{\mathbf{G}}_1$. Geometry then follows

$$\mathbf{G}_{k+1} = \mathbf{G}_k + h\frac{\widetilde{\mathbf{G}}_1 - (1-\sigma_{\min})\mathbf{G}_k}{s_{t_k}}. \tag{9}$$

For categorical field r, a masked slot jumps to a real class a with probability

$$\Pr\Big(z_{t_{k+1},i}^{(r)} = a \;\Big|\; z_{t_k,i}^{(r)} = m_r, \mathbf{X}_{t_k}\Big) = \frac{1}{K-k}\, p_\theta\Big(z_{1,i}^{(r)} = a \mid \mathbf{X}_{t_k}, t_k\Big). \tag{10}$$

The remaining probability leaves the field masked. Once revealed, a categorical value is copied unchanged, and all categorical fields are revealed by the final update. The updated mixed state is fed directly into the next step. The lower half of Figure 2 illustrates the joint trajectory.

Because $\sigma_{\min} > 0$ leaves residual noise at $t = 1$, we make one additional network call for terminal denoising. The same type-probability gate is applied to its geometry prediction, after which each final type selects the active geometry block of its slot.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Experimental data. Following SketchDNN, we merge the original SketchGraphs partitions, retain sketches containing 7–16 primitives, and create a seeded $90/5/5$ percent train/validation/test split. Every method is trained and evaluated on this shared split.

Model architecture. JFM, CFM, and DFM denote the three path allocations and share the same 32-block, width-512, eight-head Transformer backbone, 1,000-epoch training recipe, and sampler defaults; only the path allocation and the required output heads differ. PARASKETCHFM is the regularized JFM configuration. SketchDNN, Vitruvion, and UniSketch are retrained on the shared split using their released configurations, with UniSketch adapted to the four-type, constraint-free observation space. Training and evaluation run on a node with eight NVIDIA H20 GPUs; Appendix B gives the complete configurations.

Evaluation metrics. We compare 10,000 generated sketches with the first 10,000 test sketches using the SketchDNN renderer and 2048-dimensional InceptionV3 features (Szegedy et al., 2016). FID measures the discrepancy between the two feature distributions (Heusel et al., 2017). Precision is the fraction of generated samples lying on the test manifold and recall the fraction of test samples lying on the generated manifold, both estimated from $k = 3$ nearest neighbors (Kynkäänniemi et al., 2019). We also report primitive-type frequencies over all 16 slots. Sampling time is the CUDA-event average over 1,000 sketches at batch 256 on one H20 GPU, excluding rendering and feature extraction. Appendix B.1 gives the full protocol, under which no generated sketch is filtered, repaired, or resampled.

4.2 GENERATION QUALITY AND DISTRIBUTIONAL FIDELITY

Table 1: Generation quality and sampling cost. All scores use the shared evaluation protocol. AR entries count sequential decoding steps; flow NFE includes terminal denoising.

Method	NFE	ms/sketch ↓	FID ↓	P ↑	R ↑
Vitruvion	AR ≤ 130	21.45	22.03	<u>0.509</u>	0.254
SketchDNN	1,999	394.96	12.05	0.338	0.441
UniSketch (ported)	AR ≤ 17	**1.34**	82.85	0.369	0.165
Continuous-all (CFM)	101	<u>19.56</u>	<u>8.30</u>	0.402	<u>0.463</u>
Discrete-all (DFM)	100	20.73	169.49	**0.549**	0.006
PARASKETCHFM (ours)	101	20.62	**5.22**	0.431	**0.490**

Quality and coverage. Among the methods we evaluate, PARASKETCHFM achieves the best FID (5.22) with balanced precision (0.431) and recall (0.490). DFM reaches precision 0.549 but recall only 0.006, an extreme form of the line-favoring direction of Section 1; Vitruvion shows the same tendency more mildly. Under the shared protocol, CFM improves the FID of our SketchDNN reproduction from 12.05 to 8.30 and brings primitive frequencies closer to the test distribution (Table 2), so the flow-matching counterpart improves both generation quality and distributional fidelity in this setting. Figure 3 shows the same pattern: DFM concentrates on rectilinear sketches, whereas CFM and JFM generate more varied shapes. UniSketch's weaker result may partly reflect its adaptation to this restricted observation space. PARASKETCHFM reaches this quality with 101 network evaluations, 19.2× faster per sketch than our SketchDNN reproduction; the ported UniSketch is faster still, at an FID of 82.85.

Table 2: Primitive-type frequencies across all slots (%). Train and Test each use 10,000 sketches; each method generates 10,000. NONE denotes unused slots.

Source	Line	Circle	Arc	Point	None
Train	48.98	4.26	4.33	6.06	36.37
Test	49.02	4.18	4.40	5.97	36.44
Vitruvion	54.67	2.93	0.55	3.32	38.53
SketchDNN	47.69	7.64	6.24	9.28	29.15
UniSketch (ported)	55.67	3.60	3.73	4.44	32.56
Continuous-all (CFM)	49.16	4.02	4.29	6.00	36.53
Discrete-all (DFM)	99.78	0.00	0.03	0.14	0.05
PARASKETCHFM (ours)	49.33	4.08	4.35	5.86	36.37

Primitive composition. Table 2 clarifies the distributional differences behind these metrics. Vitruvion and the UniSketch port generate lines more frequently than the test set, while SketchDNN assigns more slots to circles, arcs, and points and fewer to NONE. CFM and PARASKETCHFM both closely reproduce the test frequencies, with maximum per-type deviations of 0.16 and 0.31 percentage points. CFM's slightly closer marginals but higher FID show that primitive frequencies are a useful diagnostic rather than a complete measure of quality.

4.3 CONTROLLED ANALYSIS

Path allocation and regularization. The advantage of the joint configuration over CFM is conditional on geometric regularization: without the pair term, the measured JFM and CFM FIDs are 23.99 and 8.18. Figure 4a,b varies η independently for both models and reports the tiny-box rate, the fraction of sketches whose final bounding-box diagonal falls below 0.1. Full P/R results are in Appendix B.3.

CFM is nearly insensitive to η: without regularization its tiny-box rate is already only 0.36%, leaving little room for the pair term to help, and its FID stays between 8.18 and 8.30. JFM starts with a much higher tiny-box rate of 2.37%. Increasing η to 10^{-3} reduces this rate to 0.49% and improves FID from 23.99 to 5.22. At $\eta = 10^{-2}$ the tiny-box rate falls further to 0.10%, but FID rises to 12.58,

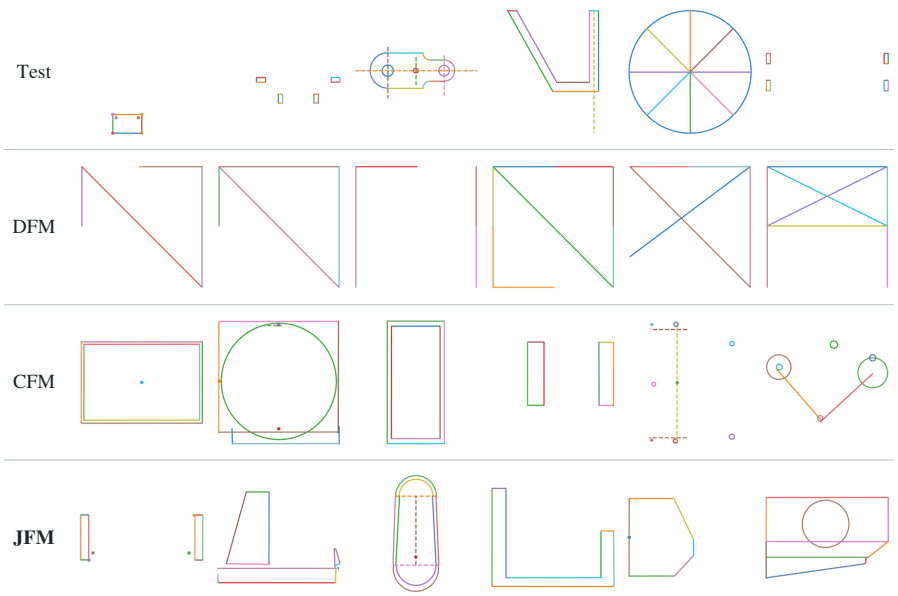

Figure 3: Random test and generated sketches. Rows show held-out test sketches, DFM, CFM, and JFM, separated by horizontal rules. Each row contains six randomly sampled or generated sketches, without individual sample selection.

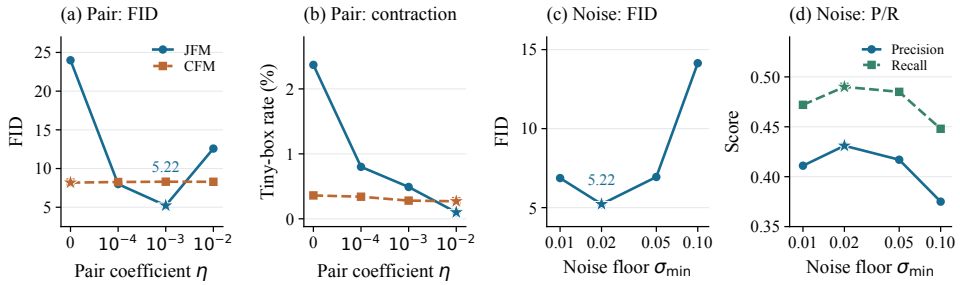

Figure 4: Hyperparameter sensitivity. (a,b) Pair regularization: FID and tiny-box rate. (c,d) Endpoint noise: FID and precision/recall. Stars mark the best observed value per curve.

suggesting that excessive separation distorts the geometric scale distribution. Thus, $\eta = 10^{-3}$ best balances contraction control and scale diversity.

Endpoint-noise sensitivity. For each $\sigma_{\min} \in \{0.01, 0.02, 0.05, 0.10\}$, we train a separate model under the shared protocol. Figure 4c,d reports FID and P/R. FID is best at $\sigma_{\min} = 0.02$ and deteriorates to 14.14 at 0.10: a small $\sigma_{\min}$ amplifies endpoint-prediction errors through $1/s_t$ near $t = 1$, whereas a large one leaves more residual noise for the single terminal-denoising call (Appendix B.5).

Sampling components. Table 3 compares sampling choices at one fixed checkpoint. Omitting inactive-block input projection improves FID from 5.63 to 5.22: the model is trained on full-dimensional noisy geometry, so zeroing unselected blocks only at sampling time introduces an input mismatch. Rescaling the gate by its maximum type probability, as SketchDNN does at inference (Chereddy & Femiani, 2025), gives FID 6.39; the unnormalized posterior keeps the weighting consistent with Equation 7.

Table 3: Sampling component comparisons. All rows share one checkpoint. The last two rows are ablations of the projection-enabled configuration, not of the no-projection default.

Sampler	NFE	FID ↓	P ↑	R ↑
Default (no inactive-block projection)	101	**5.22**	**0.431**	0.490
Max-normalized gating	101	6.39	0.417	0.471
With input projection	101	5.63	0.429	**0.494**
Projection, without gating	101	17.99	0.317	0.453
Projection, without terminal denoising	100	44.31	0.228	0.203

Removing the gate or the terminal denoising call degrades FID much further, to 17.99 and 44.31, because the geometry blocks then lose their type weighting or keep the residual noise that a positive $\sigma_{\min}$ leaves behind.

4.4 Generation dynamics

Geometry around type revelation. Figure 5 follows four slots from the lower trajectory in Figure 2. Coordinates outside each final active block approach zero, as does the None slot. Within these slots the geometry changes both before and after the type is revealed. This trajectory illustrates that geometry is not produced in a separate stage once the type is fixed.

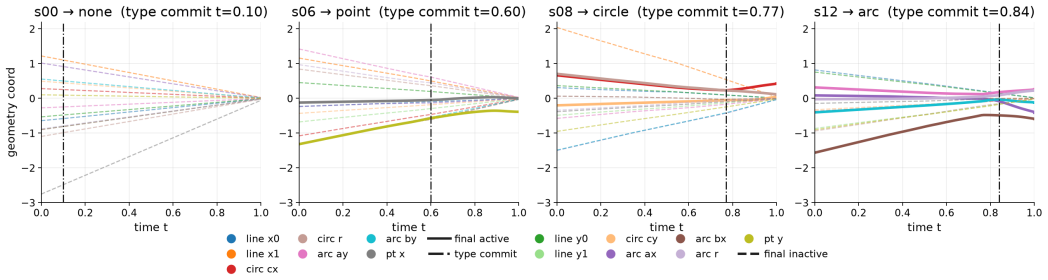

Figure 5: Slot geometry during joint generation. The four slots come from the lower trajectory in Figure 2 and end as None, point, circle, and arc. Dash-dotted vertical lines mark type commitment; solid and dashed curves denote coordinates active and inactive under the final type.

4.5 Scope and limitations

Our model generates primitive geometry and construction labels but does not model geometric constraints between entities. Consequently, strong generation quality and distributional fidelity do not guarantee constraint solvability, closed profiles, or manufacturability.

Our formulation also assumes a fixed number of slots, four primitive types, and a zero-padded per-type geometry layout. Type-probability gating and the anchor-distance regularizer are designed for this layout; richer vocabularies, variable-length sets, or other parameter layouts may need different coupling rules and geometric priors.

5 Conclusion

We introduced ParaSketchFM, a joint discrete–continuous flow-matching model for parametric CAD sketch generation. It addresses the two challenges raised above: primitive-type frequencies match the SketchGraphs test set to within 0.31 percentage points per type, and a one-sided anchor-distance regularizer removes the near-point contraction of the unregularized joint model. Comparing continuous-all, discrete-all, and joint paths under one representation and backbone shows the regularized joint path performs best, reaching FID 5.22 with balanced precision and recall under the shared protocol. Extending generation to geometric constraints remains future work.

AI USE STATEMENT

We used large language models for four purposes in this work. First, we used them to search for and discover related literature, broadening the coverage of our own survey; the retrieved papers were not always on topic, so we read and screened every suggested paper. Second, we used them to assist with LaTeX typesetting, which reduced repetitive formatting work. Third, we used them for language polishing: our first draft exceeded the nine-page limit, so we asked the model to condense selected paragraphs while preserving their meaning. Fourth, we used them to assist in writing code. All code was reviewed by the authors, and every experiment reported here was planned by the authors on the basis of our own analysis of earlier results.

We did not use generative AI tools to generate synthetic data, to develop the theoretical formulation or conceptual framework, to formulate or prove mathematical claims, to propose or refine our hypotheses, or to design the methodology or the experiments. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The settings needed to reproduce our results are specified in the appendix. Appendix B gives the backbone, optimizer, schedule, and sampler defaults of PARASKETCHFM and of the continuous-all and discrete-all variants, together with the reproduction settings of SketchDNN, Vitruvion, and UniSketch. The data split is described in Section 4, and the evaluation protocol in Appendix B.1, including the precision/recall estimator we use and how it differs from the released SketchDNN implementation. Appendix B.2 documents the encoding and decoding policy of the UniSketch port, Appendix B.3 the anchor definitions and filtering rules of the pair-distance regularizer, and Appendix B.5 the step and endpoint-noise sweeps. Appendix A derives the endpoint parameterization, the loss-to-velocity conversion, the frozen-endpoint Euler update, and the absorbing jump rate used by the sampler.

REFERENCES

Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. URL https://jmlr.org/papers/v26/23-1605.html.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5453–5512. PMLR, 2024. URL https://proceedings.mlr.press/v235/campbell24a.html.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. In *Advances in Neural Information Processing Systems*, 2020.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Sathvik Reddy Chereddy and John Femiani. SketchDNN: Joint continuous-discrete diffusion for CAD sketch generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10199–10215. PMLR, 2025. URL https://proceedings.mlr.press/v267/chereddy25a.html.

Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3D de novo molecule generation, 2024a. URL https://arxiv.org/abs/2404.19739.

Ian Dunn and David Ryan Koes. Exploring discrete flow matching for 3D de novo molecule generation. In *NeurIPS 2024 Workshop on Machine Learning for Structural Biology*, 2024b. URL https://arxiv.org/abs/2411.16644. Workshop paper.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/2e92962c0b6996add9517e4242ea9bdc-Abstract.html.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, pp. 133345–133385, 2024. URL https://papers.neurips.cc/paper_files/paper/2024/hash/f0d629a734b56a642701bba7bc8bb3ed-Abstract-Conference.html.

Julian Jorge Andrade Guerreiro, Naoto Inoue, Kento Masui, Mayu Otani, and Hideki Nakayama. LayoutFlow: Flow matching for layout generation. In *Computer Vision – ECCV 2024*, volume 15094 of *Lecture Notes in Computer Science*, pp. 56–72. Springer, 2024. doi: 10.1007/978-3-031-72764-1_4.

David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887. PMLR, 2022. URL https://proceedings.mlr.press/v162/hoogeboom22a.html.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10167–10176, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Inoue_LayoutDM_Discrete_Diffusion_Model_for_Controllable_Layout_Generation_CVPR_2023_paper.html.

Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. SolidGen: An autoregressive model for direct B-rep synthesis. *Transactions on Machine Learning Research*, 2023.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html.

Elad Levi, Eli Brosh, Mykola Mykhailych, and Meir Perez. DLT: Conditioned layout generation with joint discrete-continuous diffusion layout transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2106–2115, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Levi_DLT_Conditioned_layout_generation_with_Joint_Discrete-Continuous_Diffusion_Layout_Transformer_ICCV_2023_paper.html.

Jing Lin, Fazhi He, and Rubin Fan. UniSketch: A unified framework for parametric sketch generation and constraint prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23532–23540, 2026. doi: 10.1609/aaai.v40i28.39525.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Yang Liu, Daxuan Ren, Yijie Ding, Jianmin Zheng, and Fang Deng. Bidirectional query-driven generation of parametric CAD sketch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3176–3185, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024.

Wamiq Reyaz Para, Shariq Farooq Bhat, Paul Guerrero, Tom Kelly, Leonidas J. Guibas, Peter Wonka, and Niloy J. Mitra. SketchGen: Generating constrained CAD sketches. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5077–5088, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/28891cb4ab421830acc36b1f5fd6c91e-Abstract.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICML 2020 Workshop on Object-Oriented Learning*, 2020. URL https://arxiv.org/abs/2007.08506.

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P. Adams. Vitruvion: A generative model of parametric CAD sketches. In *International Conference on Learning Representations*, 2022.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

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.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

Clément Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. MiDi: Mixed graph and 3D denoising diffusion for molecule generation. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, volume 14170 of *Lecture Notes in Computer Science*, pp. 560–576. Springer, 2023. doi: 10.1007/978-3-031-43415-0_33.

Karl D. D. Willis, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *CVPR Workshops*, 2021a.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Transactions on Graphics*, 40 (4):1–24, 2021b. doi: 10.1145/3450626.3459818.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 24698–24724. PMLR, 2022. URL https://proceedings.mlr.press/v162/xu22k.html.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.

Jin Zhou, Hongliang Yang, Pengfei Xu, and Hui Huang. SketchFlow: Zero-shot vector sketch generation via GMM prior flow in CLIP latent space. In *SIGGRAPH Asia 2026 Conference Papers*. ACM, 2026. doi: 10.1145/3829340.3842307.

A Derivations for the continuous path, objective, and sampler

This appendix records algebra omitted from Section 3: the velocity parameterization, the loss-to-velocity conversion, the effect of per-sketch normalization on the endpoint estimator, frozen-endpoint Euler integration, and the absorbing jump rate. The geometry and pair-distance objectives are stated in Section 3.3.

Endpoint parameterization. For the raw endpoint prediction $\widehat{\mathbf{G}}_1$, define the velocity used to relate endpoint regression to conditional velocity error:

$$\mathbf{V}_\theta(\mathbf{X}_t, t) = \frac{\widehat{\mathbf{G}}_1 - (1 - \sigma_{\min})\mathbf{G}_t}{s_t}. \tag{11}$$

The sampler instead substitutes the gated prediction $\widetilde{\mathbf{G}}_1$ into this expression. At $\sigma_{\min} = 0$, the terminal denominator vanishes, requiring a different treatment such as direct velocity prediction. We do not evaluate that parameterization.

Loss-to-velocity conversion. The conditional geometric velocity is $\mathbf{u}_t = \mathbf{G}_1 - (1 - \sigma_{\min})\mathbf{E}$. Equation 11 gives $\mathbf{V}_\theta - \mathbf{u}_t = (\widehat{\mathbf{G}}_1 - \mathbf{G}_1)/s_t$. Thus Equation 5 is a type-masked, per-sketch-weighted velocity regression after rescaling; this algebra does not make it unmasked regression of the complete marginal field. Over ideal uniform time, $\int_0^1 \sigma_{\min} s_t^{-2}\,\mathrm{d}t = 1$. Its shape changes with $\sigma_{\min}$: λ rises from $\bar{\lambda}_g \sigma_{\min}$ to $\bar{\lambda}_g/\sigma_{\min}$. Smaller positive values concentrate supervision nearer the endpoint. The implementation clips time as specified below.

Type-conditional optimum under normalization. Let $x = (\mathbf{X}_t, t)$, let Y be a coordinate in the block for type τ of slot i, and let $D = D(\mathbf{C})$ be the number of active coordinates. With $\eta = 0$ and unrestricted pointwise predictors, differentiating the conditional weighted squared error gives

$$f^*_{i,\tau}(x) = \frac{\mathbb{E}[\mathbf{1}_{\{c_{1,i}=\tau\}} Y/D \mid x]}{\mathbb{E}[\mathbf{1}_{\{c_{1,i}=\tau\}}/D \mid x]} = \frac{\mathbb{E}[Y/D \mid x, c_{1,i} = \tau]}{\mathbb{E}[1/D \mid x, c_{1,i} = \tau]}. \tag{12}$$

This holds when the conditioning type has positive probability; the positive time weight cancels at fixed x. Writing $\mu_{i,\tau} = \mathbb{E}[Y \mid x, c_{1,i} = \tau]$, the bias relative to the ordinary conditional mean is

$$f^*_{i,\tau} - \mu_{i,\tau} = \frac{\mathrm{Cov}(Y, D^{-1} \mid x, c_{1,i} = \tau)}{\mathbb{E}[D^{-1} \mid x, c_{1,i} = \tau]}. \tag{13}$$

It vanishes, for example, when D is fixed under the conditioning or this covariance is zero; we do not assume that it is uniformly small. Equation 7 is an exact identity for true probabilities and ordinary conditional means. Multiplying the reweighted optimum by an ordinary type posterior does not in general recover that identity. With $\eta > 0$, Equation 6 further changes the optimum through nonlinear interactions across anchors. Finite model capacity and shared parameters introduce additional approximation. These distinctions motivate the cautious interpretation of the gate in Section 3.4.

Frozen-endpoint Euler. With $\widetilde{\mathbf{G}}_1$ held fixed across a step, $\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{G}_t/s_t) = \widetilde{\mathbf{G}}_1/s_t^2$ integrates to exactly Equation 9. Explicit Euler is therefore exact for this affine path given a frozen endpoint; the residual error is the drift of $\widetilde{\mathbf{G}}_1$ rather than quadrature error in a known velocity field. A higher-order solver would help only by re-estimating the endpoint at intermediate times.

Absorbing jump rate. For categorical field r, a still-masked slot jumps to class a with probability

$$\Pr\Big(z^{(r)}_{t_{k+1},i} = a \;\Big|\; z^{(r)}_{t_k,i} = m_r, \mathbf{X}_{t_k}\Big) = \rho_k\, p_\theta\Big(z^{(r)}_{1,i} = a \mid \mathbf{X}_{t_k}, t_k\Big),$$

with $\rho_k = \min\{h/(1-t_k), 1\}$. For $t_k = k/K$ this reduces to $\rho_k = 1/(K-k)$, which is Equation 10, so every field is revealed by the final step. Training draws t uniformly and clips it to $[10^{-4}, 1-10^{-4}]$; the sampler nevertheless evaluates $t = 0$ and $t = 1$. The extrapolation distance is 10^{-4}. Because $\sigma_{\min} > 0$, the ODE endpoint is still noisy, and the gated terminal call denoises that state rather than extending the numerical integration.

B EXPERIMENTAL DETAILS

Main model and shared protocol. A clean slot has 21 channels: 14 geometric coordinates, two construction classes, and five primitive types; adding one `[MASK]` state per categorical field gives a 23-dimensional model input. The backbone is a 32-block, width-512, eight-head DiT with AdaLN-Zero. JFM is trained for 1,000 epochs with AdamW (Loshchilov & Hutter, 2019), effective batch size 8,192, zero weight decay, and gradient clipping at 1.0. The learning rate warms up for ten epochs to 10^{-4} and follows cosine decay to 10^{-5}. EMA decay is 0.9995. Training draws t uniformly and clips it to $[10^{-4}, 1 - 10^{-4}]$ for numerical stability. The default configuration uses the normalized CFM weight of Equation 5 with $\bar{\lambda}_g = 6$, $\sigma_{\min} = 0.02$, and pair coefficient $\eta = 10^{-3}$. Unless an ablation explicitly changes the sampler, evaluation uses the epoch-1,000 EMA checkpoint, unit temperature, 100 Euler steps, terminal denoising, and no inactive-block input projection. Pair and endpoint-noise settings are trained independently; the step sweep changes only the number of Euler steps.

Path variants. Continuous-all and discrete-all use the same data, 32-block width-512 backbone, optimizer, 1,000-epoch schedule, EMA checkpoint, sampling budget, no-projection sampler, and evaluation protocol as the main JFM experiment. Only the probability-path allocation and the output heads change. Continuous-all keeps the geometric flow and sends both categorical fields through Gaussian-source flows to centered logits with label smoothing $k = 0.99$; discrete-all quantizes active geometry and generates geometric bins and attributes alike with absorbing discrete flows. Continuous-all uses the same pair coefficient as JFM. Discrete-all has no Euclidean anchors and therefore no pair term. Output-head dimensions differ between representations because discrete-all predicts a bin distribution per geometry coordinate.

Sampling configurations. All rows of Table 3 use the same epoch-1,000 EMA checkpoint. The default omits input projection. The max-normalized row keeps that sampler and only divides $\mathbf{p}_i^{\text{type}}$ by its largest entry before gating, so the block selected by the arg-max type always receives unit weight. The projection row adds inactive-block input projection, and the final two rows disable gating or terminal denoising relative to that projection-enabled row rather than to the default. In the ungated row, $\widehat{\mathbf{G}}_1$ replaces $\widetilde{\mathbf{G}}_1$ in Equation 9; the no-terminal row decodes the last integrated state at 100 rather than 101 evaluations.

Reproduced baselines. No public checkpoint was used for the three prior generators. We retrain each model on the shared split, generate 10,000 samples, and evaluate them with the same renderer, feature extractor, and reference cache as JFM. SketchDNN reports FID 7.80 in its own evaluation (Chereddy & Femiani, 2025), which uses its own data preparation and reference set; FID depends on the reference distribution, so published values are not directly comparable with Table 1. The model-specific reproduction settings are as follows.

- **SketchDNN.** We use the released 32-layer, width-512, eight-head set Transformer, Adam with learning rate 10^{-4}, 1,000 epochs, effective batch size 4,096, and the released continuous and categorical diffusion objectives. Sampling follows the 2,000-step diffusion schedule, which makes 1,999 reverse network calls.
- **Vitruvion.** We train the unconditional primitive generator: a 12-layer, width-256 decoder-only Transformer with eight heads. We use AdamW, the released one-cycle schedule initialized at 3×10^{-5} for reference batch 128, 30 epochs, and effective batch size 4,096. Continuous parameters are quantized to six bits. Generation is autoregressive and stops at the end token or the 130-token limit.
- **UniSketch.** We use the sequence-only entry point with no image conditioning: six decoder blocks, width 512, eight heads, feed-forward width 512, and dropout 0.1. The port is trained for 60 epochs with Adam, learning rate 10^{-3}, 2,000 warmup steps, gradient clipping at 1.0, and effective batch size 16,384. Generation uses the released autoregressive sampler, with at most 17 primitive-decoding steps in this adapted observation space.

UniSketch additionally requires the encoding and decoding adaptations in Appendix B.2; its row is an architecture port rather than a reproduction of the published seven-primitive, constraint-generating task. Figure A1 shows samples from the three reproductions.

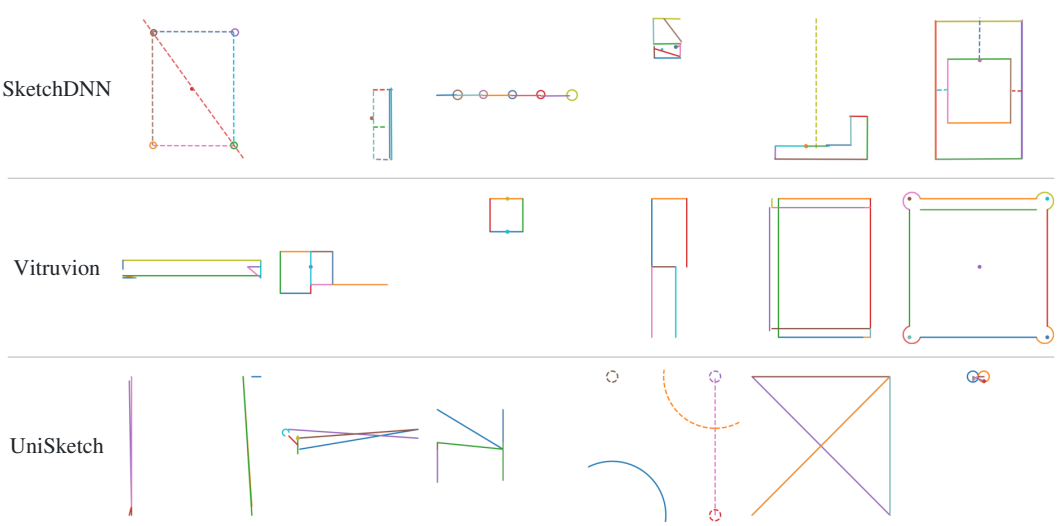

Figure A1: Qualitative samples from reproduced prior generators. Rows show SketchDNN, Vitruvion, and the UniSketch architecture port, with six supplied samples per model. Horizontal rules separate models; sample order is preserved.

B.1 Evaluation protocols

Precision/recall implementation. Given L_2-normalized features, both implementations we discuss declare a sample covered when its distance to some opposite-set feature falls below 0.95 times that feature's manifold radius, and both set the radius from $k = 3$ neighbors. They differ only in how the self-distance is removed. The released SketchDNN notebook computes the radius as

```
np.mean(np.partition(D, k+1, axis=1)[:, 1:k+1], axis=1)
```

on the full pairwise distance matrix `D`, which discards column 0 after an introselect partition. That is correct only if the zero self-distance lands in column 0. `np.partition` guarantees that the element at the pivot index is in sorted position and that all preceding elements are no larger, but it does not order those preceding elements, so the self-distance can survive into the selected neighbors while a genuine neighbor is dropped. Whenever that happens the radius is underestimated, both manifolds shrink, and precision and recall fall together. We instead set the diagonal to $+\infty$ before taking the k smallest distances, which removes the self-distance by index rather than by position.

We generate the samples for every row of Table 1 ourselves and therefore apply the corrected estimator uniformly. Table A1 applies both estimators to one fixed archived feature set, with features, renderer, reference set, and threshold held constant, so the difference between the two rows is an artifact of the radius estimator alone. That archived set is not the Table 1 evaluation. The published values, produced with the literal estimator, are listed in the second block.

Table A1: Precision and recall under the two neighborhood estimators. The first block applies both estimators to one archived 10,000-sample ParaSketchFM feature set, so its rows differ only in the radius estimator; that set is not the Table 1 evaluation. The second block lists published values, obtained with the literal estimator, and is comparable with the first row only. Best values per column are bold within each block.

Source and estimator	P ↑	R ↑
ParaSketchFM, literal	0.259	0.303
ParaSketchFM, corrected	**0.419**	**0.487**
SketchDNN (published)	0.246	**0.266**
Vitruvion (published)	**0.294**	0.176

Under the common literal estimator, PARASKETCHFM improves on the published SketchDNN precision and recall by 0.013 and 0.037 respectively, and exceeds Vitruvion's recall while remaining below its precision. The two estimators are not interchangeable across sources: reading our corrected values against the published literal ones would substantially overstate the improvement.

On likelihoods. Vitruvion scores quantized ordered token sequences and the released SketchDNN evaluator computes an SNR-weighted reconstruction proxy, so neither quantity shares an observed variable, base measure, or ordering convention with the set-based continuous model evaluated here. We therefore report sample-based metrics only.

Internal comparisons share the renderer, Inception implementation, reference features, and no-projection sampler. Each noise setting uses an independently trained epoch-1,000 EMA checkpoint. Colored figures serve visualization; distribution metrics use monochrome renderings.

B.2 THE UNISKETCH PORT

The UniSketch row of Table 1 is not a reproduction of the published UniSketch result. It is the UniSketch architecture trained on the observation space of this paper: the same preprocessed SketchGraphs tensors, the same four real primitive types plus NONE, and no constraints. The mapping between the two representations is lossy in both directions, so this section states the encoding and decoding policy in full.

Source data and command encoding. Inputs are the same $[N, 16, 21]$ tensors used by every other row, containing only LINE, CIRCLE, ARC, POINT, and NONE; the source corpus contains no ellipses, elliptical arcs, or conics, and no relations. Each sketch is converted to a UniSketch command sequence by skipping NONE slots, so sequences are variable-length with between seven and sixteen primitives rather than a fixed sixteen-slot superposition. Continuous parameters are quantized to 64 uniform bins. A line is encoded by its two endpoints and direction angle, a circle by center and radius, a point by its coordinates, and an arc by first resolving SketchDNN's three-point parameterization into center, radius, and start and end angles. The construction flag is carried in the command's second argument. Primitive type indices are remapped, since the two codebases order the classes differently: SketchDNN uses LINE $= 0$, CIRCLE $= 1$, ARC $= 2$, POINT $= 3$, NONE $= 4$, whereas UniSketch uses POINT $= 0$, LINE $= 1$, CIRCLE $= 2$, ARC $= 3$.

Retained vocabulary and the constraint head. We keep UniSketch's full twelve-command vocabulary, including the three primitive classes absent from this corpus, the selection command, the constraint commands, and the sequence delimiters. The model can therefore emit symbols that never occur in its training data. Likewise, each training sequence terminates immediately after its primitives, so the constraint head receives supervision only on an empty constraint suffix while still contributing to the loss at its default weight. This deviates from UniSketch's intended configuration and spends capacity and gradient on a degenerate target. The sequence-only training configuration is given at the start of this appendix.

Decoding policy. Rendering a generated command sequence with SketchDNN's renderer requires a fixed policy for symbols outside this observation space. We apply the following rules, in order, to every generated sequence without exception and without quality-based filtering.

1. Every token after the end-of-primitive delimiter, and every selection or constraint command wherever it occurs, is discarded. Constraints are outside the output space of all compared models and are never rendered.
2. An emitted ellipse, elliptical arc, or conic is discarded together with its arguments. These classes have no counterpart among the four types the renderer accepts, and mapping them to a nearest neighbor would require an arbitrary geometric projection. The sketch is kept; only the offending primitive is dropped.
3. A sequence yielding more than sixteen primitives is truncated to the first sixteen in emission order; a sequence yielding fewer than seven is kept as generated. We do not discard low-cardinality samples, because doing so would filter the baseline's output on a quality-correlated criterion. Remaining slots are filled with NONE, returning the sample to the $[16, 21]$ layout the renderer expects.

4. Bin index $k \in \{0, \dots, 63\}$ for a parameter with range $[\ell, u]$ decodes to $\ell + (k + \frac{1}{2})(u - \ell)/64$. Coordinates use $[-1, 1]$, radii $[0, 2]$, and angles $[0, 2\pi)$. Arcs are converted back from center, radius, and angles to the three-point form the renderer consumes.

What the row does and does not show. The comparison fixes the data split and final evaluation observation space, renderer, and metrics. It compares adapted systems, including their representation and decoding policies, rather than isolating architecture alone. It does not evaluate UniSketch's own contribution, which is joint primitive and constraint modeling over a seven-primitive vocabulary that this corpus does not exercise.

B.3 PAIR-DISTANCE REGULARIZER

This section specifies the quantities used in Equation 6. For each slot, the anchor extractor $A(\mathbf{g}_i, c_i)$ returns two line endpoints, four axis-extreme points for a circle, two arc endpoints, one point coordinate, or no anchors for NONE, and ground-truth types select both target and predicted anchors. We draw 64 anchor-index pairs independently and retain those for which both anchors are valid and the indices differ, giving the multiset $\mathcal{P}$. The distance is floored to avoid a singular logarithm, $d_\epsilon(\mathbf{a}, \mathbf{b}) = \sqrt{\max(\|\mathbf{a} - \mathbf{b}\|_2^2, 10^{-2})}$, and the margin is $\mu = 0.05$. Table A2 reports the coefficient sweep.

Without the regularizer the two continuous-geometry variants are ordered the other way (23.99 against 8.18). Raising η reduces the tiny-box rate for PARASKETCHFM from 2.37% to 0.10%, while continuous-all already starts at 0.36% and moves only slightly. The pair term is therefore not a uniform improvement applied on top of an existing advantage: PARASKETCHFM is the more sensitive of the two to the scale collapse the term targets, and also the one that benefits far more once it is available.

Table A2: Pair-distance regularization. Each coefficient is trained independently under the shared no-projection protocol and evaluated with one generation seed and 10,000 samples. Tiny-box denotes the percentage of sketches with extent below 0.1.

Variant	η	FID ↓	P ↑	R ↑	Tiny-box (%) ↓
Continuous-all (CFM)	0	8.18	0.390	0.464	0.36
Continuous-all (CFM)	10^{-4}	8.26	<u>0.405</u>	0.465	0.34
Continuous-all (CFM)	10^{-3}	8.30	0.402	0.463	0.28
Continuous-all (CFM)	10^{-2}	8.29	<u>0.405</u>	<u>0.468</u>	<u>0.27</u>
PARASKETCHFM	0	23.99	0.329	0.287	2.37
PARASKETCHFM	10^{-4}	<u>7.99</u>	0.394	0.454	0.80
PARASKETCHFM	10^{-3}	**5.22**	**0.431**	**0.490**	0.49
PARASKETCHFM	10^{-2}	12.58	0.352	0.451	**0.10**

B.4 ADDITIONAL ABLATIONS

Primitive order and source semantics. NONE denotes a terminal empty slot, whereas `[MASK]` denotes an unresolved categorical field. We use a separate mask source to distinguish these roles and omit positional encoding to preserve slot-permutation equivariance.

Type-posterior sharpening. Type-probability gating scales each predicted geometry block. Figure A2c plots the mean maximum type probability and the mean sketch extent along the default trajectory, describing posterior sharpening as sampling proceeds.

Permutation-equivariance check. For a random slot permutation matrix P, we compare $F_\theta(P\mathbf{x}_t, t)$ with $PF_\theta(\mathbf{x}_t, t)$ separately for geometry and both categorical logits. Sampling is checked by permuting the initial Gaussian and categorical random sources, applying the same random draws to both runs, and undoing the output permutation. The acceptance tolerance is a maximum absolute deviation of 10^{-5} in FP32; the model-output and decoded-sample deviations are recorded separately. This property makes reordering a fixed observed subset immaterial.

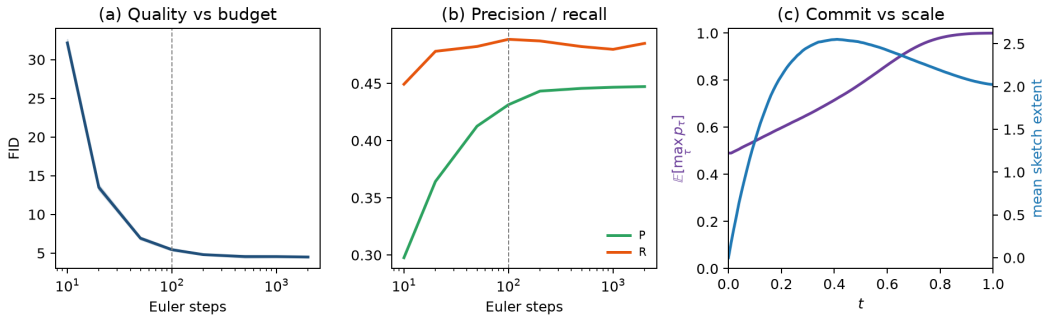

Figure A2: Sampling budget and trajectory diagnostics. (a) FID and (b) precision and recall versus Euler steps; dashed lines mark 100 steps. (c) Mean maximum type probability and mean sketch extent along a trajectory.

B.5 Step and endpoint-noise sweeps

Table A3 gives the endpoint-noise values behind Figure 4c,d, and Table A4 the sampling-budget sweep.

Table A3: Sensitivity to endpoint noise. Each row is independently trained and evaluated without input projection; the $\sigma_{\min} = 0.02$ row is the main result from Table 1.

$\sigma_{\min}$	FID ↓	P ↑	R ↑
0.01	6.87	0.411	0.472
0.02	**5.22**	**0.431**	**0.490**
0.05	6.94	0.417	0.485
0.10	14.14	0.375	0.448

Each endpoint-noise setting uses a separately trained epoch-1,000 EMA checkpoint. All sweeps use the shared no-projection sampler with terminal denoising.

Figure A2 summarizes the sampling-budget sweep. FID improves sharply up to about 100 steps and then flattens: 6.92 at 50 steps, 5.22 at 100, 4.80 at 200, 4.55 at 500, and 4.49 at 2,000. These diminishing returns motivate the 100-step operating budget.

Table A4: Sampling-budget sweep at $\sigma_{\min} = 0.02$ under the shared no-projection protocol.

Steps	NFE	FID ↓	P ↑	R ↑
50	51	6.92	0.413	0.482
100	101	5.22	0.431	**0.490**
200	201	4.80	0.443	0.487
500	501	4.55	0.446	0.482
2000	2001	**4.49**	**0.447**	0.485