EFFICIENT POINT CLOUD UPSAMPLING VIA LOCAL FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative point cloud upsampling becomes increasingly expensive as the requested output point count grows. We present FLUSSO, a flow-matching framework that separates sparse global geometric reasoning from dense local generation. A sparse encoder extracts global context, while a shared velocity field evolves particles within adaptive anchor-centered supports. To train these local flows, *sliced optimal-transport flow matching* constructs candidate correspondences through one-dimensional projections, scores their transport costs in three dimensions, and softly weights multiple low-cost coupling losses. Complementary *multi-step local endpoint distribution matching* supervises the terminal geometry reached by differentiable model rollouts. At inference, local predictions are merged and subsampled when necessary, allowing one model to support different output cardinalities. Experiments on PU1K and PUGAN demonstrate competitive reconstruction quality and favorable latency and memory efficiency under the evaluated protocols. At $16\times$ upsampling on PU1K, FLUSSO reduces squared Hausdorff and point-to-surface errors by 42.0% and 26.0%, respectively, compared with PUFM evaluated using the same metric pipeline. A model trained at $4\times$ also generalizes to the evaluated unseen ratios from $5\times$ to $32\times$ without retraining.

1 INTRODUCTION

Point cloud upsampling aims to recover dense, uniform, and geometrically faithful surface samples from sparse 3D observations. Unlike image super-resolution, the output is an unordered point set whose quality depends on accurate surface reconstruction together with coherent coverage and sampling regularity (Liu et al., 2026). Generative approaches transform sparse observations into dense point distributions using normalizing flows, diffusion, or flow matching (Mao et al., 2023; Qu et al., 2024; Zhang et al., 2025; Du et al., 2025; Liu et al., 2026). Iterative denoising and flow integration become increasingly expensive when the number of points updated at each step grows with the requested output size. However, the sparse input already provides much of the global geometry needed to guide densification, while newly generated points primarily refine local surface regions.

In addition to the computational burden of dense generation, flow matching for unordered point sets presents two challenges. First, conditional flow matching requires a source–target coupling to specify intermediate states and target velocities (Tong et al., 2024), but unordered source particles and dense target samples admit no canonical pointwise correspondence. Unconstrained pairings may induce unnecessarily long transport trajectories, whereas computing a multidimensional optimal transport (OT) coupling with standard discrete solvers for each local region requires dense pairwise costs and iterative or combinatorial optimization (Pooladian et al., 2023). Second, velocity supervision is imposed on analytically constructed intermediate states, whereas inference evaluates the learned field along its own numerically integrated trajectories. Errors in the predicted velocity may therefore accumulate across integration steps, causing the final generated point distribution to deviate from the target local surface distribution. These limitations motivate transport-informed correspondence construction and supervision of the model-induced terminal distribution.

We therefore view point cloud upsampling as *locally generative but globally conditioned* and propose flow-based local upsampling of sparse surface observations (FLUSSO), *i.e.* a flow-matching framework that performs global geometric reasoning at the sparse input resolution while decomposing dense generation into compact local flows. Generation anchors selected from the sparse cloud

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.

each define an adaptive metric support in which source particles are initialized and evolved by a shared geometry-conditioned velocity field. Global context is extracted from the sparse input, while high-resolution particle interactions remain confined to the local supports.

To address the correspondence problem, we introduce *Sliced Optimal-Transport Flow Matching (SOT-FM)*. Randomized low-discrepancy projections produce candidate matchings through one-dimensional OT (Rabin et al., 2011; Nguyen et al., 2024); each matching is evaluated using its transport cost in the original 3D space (Mahey et al., 2023). Several low-cost couplings are retained, each defining a conditional flow-matching path for the local point configuration (Lipman et al., 2023; Tong et al., 2024); the corresponding velocity losses are softly weighted. To reduce the mismatch between analytical paths and model-induced dynamics, we introduce *Multi-Step Local Endpoint Distribution Matching (LEDM)*. It differentiates through a multi-step rollout from the local source particles and matches the resulting terminal distribution to the dense local target. At inference, the evolved local particles are mapped to a shared coordinate frame and merged into a candidate point cloud. Global farthest point sampling (FPS) selects the requested number of points when the candidate count exceeds the requested output cardinality; when the counts match, all candidates are retained. Adjusting the particle count per anchor allows the same velocity field to support standard, high, and unseen upsampling ratios without retraining or recursive application of a fixed-ratio model. Experiments assess reconstruction, unseen-ratio generalization, efficiency, and noise robustness on PU1K and PUGAN, with qualitative transfer to KITTI (Geiger et al., 2013) and ScanNet (Dai et al., 2017). Our main contributions are:

- We propose a flow-matching framework that separates global geometric reasoning on the sparse input from compact local generation, avoiding global attention over dense candidates while supporting different upsampling ratios with shared parameters.
- We introduce SOT-FM to address the lack of canonical correspondences between unordered local source and target sets. It supervises the local velocity field using multiple sliced transport plans selected by their 3D costs, while preserving separate source–target correspondences.
- We develop LEDM to address the mismatch between analytically supervised paths and model-induced trajectories. It complements flow-matching supervision by matching the terminal local distribution of differentiable multi-step rollouts to dense targets.

2 Related Work

Point cloud upsampling. Learning-based point cloud upsampling has evolved from direct point expansion toward geometry-aware, flexible-scale, and generative formulations. Early methods enlarge sparse point sets through hierarchical feature expansion or adversarial generation (Yu et al., 2018; Li et al., 2019; Qian et al., 2021), while subsequent approaches incorporate stronger local geometric priors, including explicit surface modeling and representative kernel deformation (Qian et al., 2020; Rong et al., 2024). Transformer-based architectures have also been explored to improve long-range feature interaction (Qiu et al., 2022). A separate line of work targets flexible output cardinalities. PU-EVA (Luo et al., 2021) and Meta-PU (Ye et al., 2022) introduce scale-adaptive generation mechanisms, while implicit representations and dual back-projection further support arbitrary upsampling ratios (Zhao et al., 2022; Liu et al., 2023). Grad-PU (He et al., 2023) formulates arbitrary-rate upsampling as iterative optimization with a learned distance function, and self-supervised or operator-based formulations have also been studied (Liu et al., 2024; Xiao et al., 2026). More recently, generative models cast upsampling as a transformation between sparse and dense point distributions. PU-Flow (Mao et al., 2023) adopts normalizing flows, PUDM (Qu et al., 2024) and PDANS (Zhang et al., 2025) use diffusion-based generation, and SuperPC (Du et al., 2025) develops a unified diffusion formulation for multiple point-cloud generation tasks. PUFM (Liu et al., 2026) instead applies flow matching to learn continuous sparse-to-dense dynamics with a small number of integration steps. PUFM++ (Liu et al., 2025b) extends this approach with two-stage training, adaptive time scheduling, and on-manifold sampling constraints. FLUSSO focuses on the computational separation of sparse global encoding and dense local dynamics, using a shared local field across output cardinalities.

Optimal transport and flow matching. Flow matching learns a time-dependent vector field along prescribed probability paths (Lipman et al., 2023). The coupling between source and target samples strongly influences the geometry of these paths. OT-based conditional flow matching therefore uses transport-informed couplings to obtain shorter and simpler trajectories than independent pairing

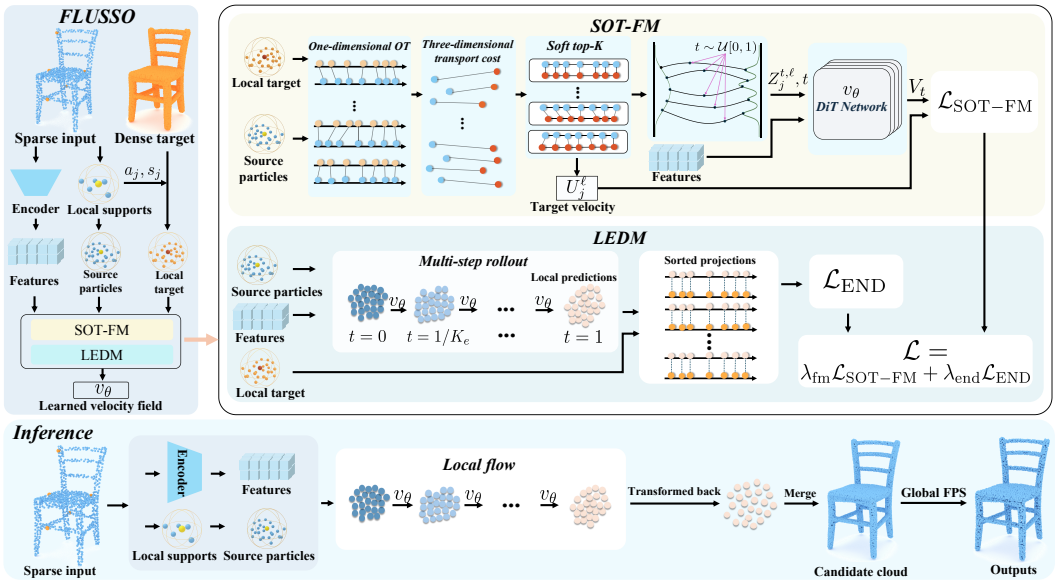

Figure 1: **Framework of FLUSSO.** Sparse global features condition shared local flows within adaptive anchor-centered supports. Training combines sliced OT flow matching with multi-step endpoint distribution matching. At inference, local predictions are merged in the shared coordinate frame; global FPS is used only when candidates exceed the requested point count. Here $V_t = v_\theta(Z_j^{t,\ell}, t, c_j)$ denotes the predicted velocity matrix.

(Tong et al., 2024), while Multisample Flow Matching similarly exploits minibatch couplings to straighten learned flows (Pooladian et al., 2023). Related work has also investigated approximate transport for 3D generation (Hui et al., 2025) and Wasserstein formulations of flow matching over distributions (Haviv et al., 2025). Constructing transport repeatedly remains costly: discrete solvers use pairwise costs, and Sinkhorn approximations require iterative matrix updates (Cuturi, 2013; Sinkhorn, 1964). Sliced optimal transport offers a cheaper alternative by projecting samples onto one-dimensional directions, where optimal transport reduces to sorting and monotone matching (Rabin et al., 2011). Mahey et al. (Mahey et al., 2023) use sliced Wasserstein generalized geodesics for efficient transport construction. Expected sliced transport plans (Liu et al., 2025a) aggregate transport information across multiple projections, while differentiable generalized sliced Wasserstein plans (Chapel et al., 2025) construct differentiable high-dimensional transport approximations and apply them to sliced-OT conditional flow matching. Quasi-Monte Carlo sampling has further been introduced to improve directional coverage in 3D sliced Wasserstein computation (Nguyen et al., 2024). Building on sliced transport plans and their use in conditional flow matching, the proposed SOT-FM constructs correspondences within anchor-centered 3D supports. Our contribution is the combination of local multi-plan velocity supervision, sparse global conditioning, and differentiable rollout endpoint matching for ratio-flexible upsampling. Unlike the two-stage training of PUFM++, both local objectives are optimized jointly, with fixed-step Euler integration at inference.

3 Method

3.1 Framework Overview

Given a sparse point cloud $X = \{x_i \in \mathbb{R}^3\}_{i=1}^N$, we aim to generate an r-times denser cloud $\widehat{Y}$ with $M = rN$ points; during training, a dense reference $Y = \{y_n \in \mathbb{R}^3\}_{n=1}^M$ is available. Figure 1 summarizes the construction. We select G input points as *anchors* and initialize q particles per anchor within adaptive metric supports. A shared velocity field evolves these particles, conditioned on features of the complete sparse cloud. All point counts are positive integers, with $1 \le G \le N$. The local predictions form a candidate point cloud $\widetilde{Y}$ with $R = qG \ge M$ points. Global FPS selects M outputs when $R > M$; otherwise, all candidates are retained as $\widehat{Y}$ (Sec. 3.6).

Throughout the method, coordinates are expressed in the globally normalized input frame. Let $\mu_X \in \mathbb{R}^3$ and $\sigma_X > 0$ denote the center and scale computed from the sparse input, so that $x^{\mathrm{phys}} = \mu_X + \sigma_X x$ relates a normalized point x to its original physical coordinates. The same transformation is applied to the dense reference during training.

3.2 Adaptive Local Support and Source Initialization

Sparse inputs and dense targets have different sampling densities, so neighborhoods with equal point counts may cover different physical regions. We instead use a common metric support for conditioning and supervision. We select G anchors $A = \{a_j\}_{j=1}^{G} \subseteq X$ by FPS and define $s_j = \beta d_{K_b}(a_j, X)$, where d_{K_b} is the distance to the K_b-th nearest non-self sparse point, $1 \le K_b < N$, and $\beta > 0$ is a radius multiplier. We assume nondegenerate inputs with $s_j > 0$. The corresponding sparse and dense supports are $\mathcal{B}_j^X = \{x \in X : \|x - a_j\|_2 \le s_j\}$ and $\mathcal{B}_j^Y = \{y \in Y : \|y - a_j\|_2 \le s_j\}$. For each anchor, we work in anchor-relative coordinates and initialize q source particles as $z_{j,m}^0 = s_j \omega_{j,m}$, where $\omega_{j,m} \sim \mathrm{Unif}(\mathbb{B}^3)$ and $\mathbb{B}^3$ denotes the three-dimensional unit ball. We denote the resulting source particle set by $P_j^0 = \{z_{j,m}^0\}_{m=1}^q$ and its coordinate matrix by $Z_j^0 = (z_{j,m}^0)_{m=1}^q \in \mathbb{R}^{q\times 3}$. Training uses random samples, whereas inference uses a fixed Sobol sequence mapped uniformly to the unit ball. The corresponding positions in the shared normalized frame are $p_{j,m}^0 = a_j + z_{j,m}^0$, yielding $R = qG$ particles in total. For supervision, we sample $(\widetilde{y}_{j,m})_{m=1}^q = \mathrm{FPS}_q(\mathcal{B}_j^Y)$ and define the anchor-relative targets $z_{j,m}^1 = \widetilde{y}_{j,m} - a_j$ and $Q_j = (z_{j,m}^1)_{m=1}^q$. Here FPS_q selects q entries by farthest point sampling; for a target support with fewer than q points, we extend its FPS ordering by cyclic repetition. During training, FPS selects X from Y, so $X \subseteq Y$ and every target support is non-empty. Local particle collections preserve repeated entries, each assigned empirical mass $1/q$; repeated coordinates may therefore receive greater total mass.

3.3 Geometry-Conditioned Local Velocity Field

An EdgeConv encoder (Wang et al., 2019) first extracts local features from the N sparse points, followed by global self-attention to produce $H = [h_1, \ldots, h_N]^\top \in \mathbb{R}^{N\times C}$, where C is the sparse feature width. For anchor $a_j = x_{\iota_j}$, with input index ι_j, we define its conditioning context as $c_j = (h_{\iota_j}, \{h_i : x_i \in \mathcal{B}_j^X\})$. The q evolving particle coordinates $Z \in \mathbb{R}^{q\times 3}$ are embedded as tokens and processed by conditioned diffusion-transformer (DiT) blocks. The anchor feature h_{ι_j} and flow time t condition each block through adaptive layer normalization (AdaLN). Each block applies self-attention among the q local particles, followed by cross-attention to sparse-support features in $\mathcal{B}_j^X$. A shared output head predicts one 3D velocity per particle, yielding $v_\theta(Z, t, c_j) \in \mathbb{R}^{q\times 3}$, where θ denotes the trainable model parameters. All anchors share the same parameters, so the particle count can vary without ratio-specific prediction layers. The architecture is provided in Appendix A.

3.4 Sliced OT Flow Matching

Conditional flow matching requires a source–target coupling (Tong et al., 2024), but P_j^0 and Q_j are unordered. We therefore construct candidate couplings using one-dimensional OT. A fixed low-discrepancy set of $L_f \ge 1$ directions on $\mathbb{S}^2$, the unit sphere in $\mathbb{R}^3$, is independently Haar-rotated for each training shape, yielding $\Theta_f = \{\xi_\ell\}_{\ell=1}^{L_f}$ shared across all local patches of that shape (Nguyen et al., 2024). For direction ξ_ℓ, we project $\alpha_{j,m}^\ell = \langle \xi_\ell, z_{j,m}^0 \rangle$ and $b_{j,n}^\ell = \langle \xi_\ell, z_{j,n}^1 \rangle$. Let $\rho_{j,0}^\ell(k)$ and $\rho_{j,1}^\ell(k)$ denote the source and target particle indices, respectively, at rank k after sorting their projections, for $k = 1, \ldots, q$. Their inverses map a particle index to its rank, and $\circ$ denotes function composition. Tied projections may be ordered arbitrarily, provided each sorting permutation and its inverse use the same ordering. Equal-rank 1D OT gives $\pi_j^\ell = \rho_{j,1}^\ell \circ (\rho_{j,0}^\ell)^{-1}$. Here $\pi_j^\ell(m)$ denotes the index of the target particle matched to source particle m, for $m = 1, \ldots, q$. To index each source and its matched target by the same m, we define $\widetilde{z}_{j,m}^{1,\ell} = z_{j,\pi_j^\ell(m)}^1$. This reorders the original 3D targets without changing their coordinates, so subsequent transport costs, interpolation paths, and target velocities can be written directly for each matched pair.

Following min-SWGG (Mahey et al., 2023), each induced coupling is scored by $C_j^\ell = \frac{1}{q}\sum_{m=1}^q \|z_{j,m}^0 - \widetilde{z}_{j,m}^{1,\ell}\|_2^2$, and normalized as $\overline{C}_j^\ell = C_j^\ell / \max(s_j^2, \epsilon_s)$. Here $\epsilon_s > 0$ is a numeri-

cal floor with the units of squared normalized distance. We retain the K lowest-cost plans, with $1 \le K \le L_f$, indexed by $\mathcal{I}_j \subseteq \{1, \dots, L_f\}$, $|\mathcal{I}_j| = K$, and assign

$$w_j^\ell = \frac{\exp[-(\overline{C}_j^\ell - \overline{C}_j^{\min})/T_c]}{\sum_{h \in \mathcal{I}_j} \exp[-(\overline{C}_j^h - \overline{C}_j^{\min})/T_c]}, \quad \ell \in \mathcal{I}_j, \tag{1}$$

where $T_c > 0$ is a soft-weight temperature and $\overline{C}_j^{\min} = \min_{h \in \mathcal{I}_j} \overline{C}_j^h$. The weights are positive and sum to one over $\mathcal{I}_j$. Retaining the plans individually, rather than averaging them into a barycentric pseudo-target, preserves actual source–target pairs. Each plan is a permutation with uniform empirical marginals, which are preserved by their weighted mixture. The sliced plans need not be optimal for the full 3D transport problem. For each retained coupling, we use the standard linear conditional path (Lipman et al., 2023; Tong et al., 2024), *i.e.*, $z_{j,m}^{t,\ell} = (1-t)z_{j,m}^0 + t\tilde{z}_{j,m}^{1,\ell}$, *velocity* $u_{j,m}^\ell = \tilde{z}_{j,m}^{1,\ell} - z_{j,m}^0$. A flow time $t \sim \mathrm{Unif}([0,1))$ is sampled per training shape and shared across its anchors and retained plans. We stack the q local states $z_{j,m}^{t,\ell}$ as rows of $Z_j^{t,\ell} \in \mathbb{R}^{q\times 3}$, and the corresponding target velocities $u_{j,m}^\ell$ as rows of $U_j^\ell \in \mathbb{R}^{q\times 3}$. Conditioned on sparse context c_j, the shared velocity field is trained by

$$\mathcal{L}_{\text{SOT-FM}} = \mathbb{E}\left[\sum_{\ell \in \mathcal{I}_j} w_j^\ell \frac{1}{q} \left\| v_\theta(Z_j^{t,\ell}, t, c_j) - U_j^\ell \right\|_F^2\right]. \tag{2}$$

Here $\|\cdot\|_F$ is the Frobenius norm. The expectation averages uniformly over anchors within each sampled training shape and over the training data, source samples, randomized directions, and flow time. Each path evolves the complete local configuration in $\mathbb{R}^{q\times 3}$; the network jointly predicts its particle velocities. Slicing constructs correspondences, while the states and velocities retain their three-dimensional coordinates.

3.5 Multi-Step Local Endpoint Distribution Matching

To reduce the resulting rollout mismatch and directly constrain the unordered terminal distribution, we start from $\widehat{Z}_j^{(0)} = Z_j^0$ and perform $K_e \ge 1$ differentiable Euler steps:

$$\widehat{Z}_j^{(k+1)} = \widehat{Z}_j^{(k)} + \frac{1}{K_e} v_\theta\left(\widehat{Z}_j^{(k)}, \frac{k}{K_e}, c_j\right), \quad k = 0, \dots, K_e - 1. \tag{3}$$

Let $\widehat{z}_{j,m}$ be the 3-vector in row m of $\widehat{Z}_j^{(K_e)}$. The terminal collection $\widehat{P}_j^1 = (\widehat{z}_{j,m})_{m=1}^q$ is matched to Q_j using a finite-direction estimate of the squared sliced Wasserstein-2 discrepancy, with mass $1/q$ per entry. With an independently randomized low-discrepancy direction set $\Theta_e = \{\zeta_\ell \in \mathbb{S}^2\}_{\ell=1}^{L_e}$, with $L_e \ge 1$ (Nguyen et al., 2024), define $\widehat{e}_{j,m}^\ell = \langle \zeta_\ell, \widehat{z}_{j,m} \rangle$ and $e_{j,m}^\ell = \langle \zeta_\ell, z_{j,m}^1 \rangle$. Parenthesized indices (m) denote their sorted order. We minimize

$$\mathcal{L}_{\text{END}} = \mathbb{E}\left[\frac{1}{L_e q} \sum_{\ell=1}^{L_e} \sum_{m=1}^{q} \left(\widehat{e}_{j,(m)}^\ell - e_{j,(m)}^\ell\right)^2\right]. \tag{4}$$

The expectation averages over training shapes, anchors, source samples, and endpoint directions. The complete objective is $\mathcal{L} = \lambda_{\text{fm}}\mathcal{L}_{\text{SOT-FM}} + \lambda_{\text{end}}\mathcal{L}_{\text{END}}$, where $\lambda_{\text{fm}}, \lambda_{\text{end}} \ge 0$ are loss weights. SOT-FM evaluates the field on analytical interpolants; LEDM evaluates states reached by the field itself. The latter compares unordered endpoints without requiring a fixed particle-wise correspondence.

3.6 Candidate Generation and Global FPS

Inference uses only the sparse input. Starting from the fixed Sobol initialization, we integrate the learned velocity field for $S \ge 1$ Euler steps using the same update as Eq. 3, replacing K_e by S. The resulting terminal offsets $\widehat{z}_{j,m}^{(S)}$, given by the rows of $\widehat{Z}_j^{(S)}$, are mapped to the shared normalized coordinate frame and merged as

$$\widetilde{Y} = \biguplus_{j=1}^{G} \left\{a_j + \widehat{z}_{j,m}^{(S)}\right\}_{m=1}^{q}, \qquad |\widetilde{Y}| = R = qG. \tag{5}$$

Table 1: Quantitative comparison at $4\times$ and $16\times$ upsampling. CD/HD/P2F are reported in units of $10^{-4}/10^{-3}/10^{-3}$; lower is better. Published results are taken from Liu et al. (2026). † denotes the official PUFM checkpoint evaluated with our metric pipeline; at $16\times$, it is recursively applied twice. Best and second-best results are shown in bold and underline.

	PU1K ($4\times$)			PUGAN ($4\times$)			PU1K ($16\times$)			PUGAN ($16\times$)		
Method	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓
PU-GCN (Qian et al., 2021)	1.241	1.504	5.115	2.774	3.831	9.508	2.393	4.214	9.410	1.102	1.785	7.125
PU-GAN (Li et al., 2019)	0.682	0.632	2.792	1.541	1.391	5.420	0.361	0.773	3.538	0.869	1.746	6.757
PU-BP (Liu et al., 2023)	0.694	0.593	2.206	1.649	1.476	5.997	0.343	0.712	2.958	0.982	2.071	7.496
PUDM (Qu et al., 2024)	0.706	0.605	2.891	1.221	1.174	3.132	0.421	0.602	3.370	0.533	1.185	3.589
Grad-PU (He et al., 2023)	0.626	0.583	1.510	1.132	1.186	1.957	0.316	<u>0.552</u>	1.890	0.415	1.142	2.185
RepKPU (Rong et al., 2024)	0.566	0.619	**1.464**	1.067	1.139	1.974	0.290	0.592	<u>1.821</u>	<u>0.384</u>	1.245	2.151
PUFM (Liu et al., 2026)	0.545	<u>0.556</u>	1.770	**1.049**	**0.876**	<u>1.864</u>	0.220	0.578	1.983	**0.353**	**0.844**	2.103
PUFM†	<u>0.464</u>	0.651	1.646	**1.049**	1.190	1.868	<u>0.219</u>	0.793	2.005	**0.353**	1.248	<u>2.096</u>
FLUSSO	**0.455**	**0.460**	<u>1.482</u>	<u>1.055</u>	<u>0.919</u>	**1.731**	**0.207**	**0.460**	**1.484**	0.399	<u>0.904</u>	**1.613**

Here $\biguplus$ denotes multiset union, preserving coincident candidates. We choose q such that $qG \geq rN$. When $qG > rN$, global FPS yields $\widehat{Y} = \mathrm{FPS}_{rN}(\widetilde{Y})$ and may retain different numbers of candidates from each anchor. When $qG = rN$, all candidates are retained, giving $\widehat{Y} = \widetilde{Y}$. Larger requested ratios can be supported by increasing q. Finally, each output $\widehat{y} \in \widehat{Y}$ is returned to physical coordinates as $\widehat{y}^{\mathrm{phys}} = \mu_X + \sigma_X \widehat{y}$ for evaluation.

4 Experiments

4.1 Experimental Setup

We train FLUSSO only at $4\times$ upsampling using 256 input and 1,024 target points per patch, with $G = N/2$ anchors and $q_{\mathrm{train}} = 24$ particles per anchor. Training uses Adam for 200 epochs with a batch size of 64. Inference uses five Euler steps and $q_{\mathrm{test}} = \max(24, 2r)$ for each evaluated integer ratio r. Architecture and training settings are provided in Appendices A and B.

Datasets. Following PUFM (Liu et al., 2026), we adopt the same combined PU1K–PUGAN training data and benchmark splits. Dense surface patches are sampled from the original meshes using Poisson-disk sampling. Sparse training inputs are selected from the dense patches by FPS with a randomly initialized first point. The PUGAN and PU1K test sets contain 27 and 127 shapes, respectively, with 2,048 input points per shape. Following PUFM, we use evaluation data prepared offline according to the official PU-GAN (Li et al., 2019) data-construction protocol. The evaluator neither resamples nor reconstructs the ground-truth point clouds; it directly pairs each input with its pre-generated ground-truth by filename.

Baselines and metrics. We compare FLUSSO with PU-GCN (Qian et al., 2021), PU-GAN (Li et al., 2019), PU-BP (Liu et al., 2023), PUDM (Qu et al., 2024), Grad-PU (He et al., 2023), RepKPU (Rong et al., 2024), and PUFM (Liu et al., 2026). Published baseline scores are taken from Liu et al. (2026). To reduce discrepancies caused by metric implementations, we additionally evaluate the official PUFM checkpoint using our evaluation pipeline and denote this result as PUFM†. We report Chamfer distance (CD), squared Hausdorff distance (HD), and point-to-surface distance (P2F). Our evaluator uses squared Euclidean distances for CD and HD, and unsigned Euclidean distances for P2F; the exact definitions are given in Appendix B.3. Published scores retain their original evaluation protocols; controlled comparisons use the shared-pipeline reruns.

4.2 Results

Comparison with baselines. Table 1 compares standard $4\times$ and high-ratio $16\times$ upsampling. At $16\times$, PUFM† recursively applies its fixed-$4\times$ model twice, whereas FLUSSO uses a single generative stage and retains all generated candidates at this ratio, without global FPS.

Table 2: Inference efficiency at matched output cardinalities. F/P denotes FLUSSO/ PUFM†. Latency is mean ± standard deviation over 50 runs after 10 warm-up iterations. Speedup is t_P/t_F, where t_P is the latency of PUFM† and t_F is the latency of FLUSSO. Profiled NN GFLOPs include neural-network operations only. PUFM† uses two recursive stages for upsampling ratios of $5\times$–$16\times$ and three for $20\times$–$32\times$, whereas FLUSSO only uses one five-step stage.

Ratio	Stages F/P	Steps F/P	Params F/P (M)	Latency (ms)		Speedup	Memory F/P (MiB)	NN GFLOPs F/P
				FLUSSO	PUFM†			
$4\times$	1/1	5/5	10.4/12.4	144.5 ± 0.7	190.5 ± 0.1	$1.3\times$	371.3/563.0	1032.3/361.1
$5\times$	1/2	5/10	10.4/12.4	156.0 ± 0.6	1796.3 ± 0.6	$11.5\times$	371.3/2061.8	1032.3/1805.3
$7\times$	1/2	5/10	10.4/12.4	177.9 ± 0.8	1824.3 ± 0.8	$10.3\times$	371.3/2061.8	1032.3/1805.3
$8\times$	1/2	5/10	10.4/12.4	189.8 ± 0.8	1839.0 ± 0.7	$9.7\times$	371.3/2061.8	1032.3/1805.3
$12\times$	1/2	5/10	10.4/12.4	100.7 ± 0.2	1895.1 ± 0.9	$18.8\times$	371.3/2061.8	1032.3/1805.3
$16\times$	1/2	5/10	10.4/12.4	119.4 ± 0.6	1726.7 ± 0.7	$14.5\times$	475.4/2061.8	1343.9/1805.3
$20\times$	1/3	5/15	10.4/12.4	140.0 ± 0.6	$20{,}577.2 \pm 9.6$	$147.0\times$	579.5/8104.1	1658.1/7582.1
$24\times$	1/3	5/15	10.4/12.4	162.1 ± 0.3	$20{,}778.8 \pm 2.5$	$128.2\times$	683.6/8104.1	1975.1/7582.1
$32\times$	1/3	5/15	10.4/12.4	208.6 ± 0.4	$21{,}186.6 \pm 2.2$	$101.6\times$	891.8/8104.1	2617.0/7582.1

Among the values reported in Table 1, FLUSSO achieves the best result in seven and the second-best in four of the 12 dataset–ratio–metric combinations. At $4\times$, it leads CD/HD on PU1K and P2F on PUGAN. At $16\times$, it leads all three metrics on PU1K and achieves the lowest P2F and second-lowest HD on PUGAN. Compared with PUFM†, FLUSSO consistently reduces HD and P2F across both datasets and ratios. At $16\times$ upsampling, the HD/P2F reductions are $42.0\%/26.0\%$ on PU1K and $27.6\%/23.0\%$ on PUGAN. PUFM† retains lower CD on PUGAN, indicating some dataset dependence in average point-set discrepancy; the HD and P2F improvements indicate smaller worst-case point-set deviations and better average surface proximity under the shared evaluator. The qualitative results in Figure 2 further show better preservation of thin structures and curved boundaries, with fewer displaced samples and more coherent local coverage. Further qualitative comparisons using spatial P2F error maps on PU1K at $4\times$ upsampling are provided in Appendix E.1.

Inference efficiency across output ratios. We benchmark inference on an isolated NVIDIA GeForce RTX 4090 GPU using 2,048-point inputs, FP32 precision, and batch size 1. We report mean latency over 50 runs after 10 warm-up iterations together with peak CUDA memory allocation. Since the FLOP profiler accounts only for neural-network operations and omits custom operators, we regard end-to-end latency and peak memory as the primary efficiency measures. As shown in Table 2, FLUSSO scales substantially better than PUFM† as the requested output cardinality increases. At $4\times$, both methods use a single five-step generative stage: FLUSSO requires $144.5\,\mathrm{ms}$ and $371.3\,\mathrm{MiB}$, compared with $190.5\,\mathrm{ms}$ and $563.0\,\mathrm{MiB}$ for PUFM†, yielding a $1.3\times$ speedup and a 34.0% reduction in peak memory. For $5\times$–$16\times$, where the evaluated PUFM† protocol uses two recursive stages, FLUSSO achieves $9.7\times$–$18.8\times$ speedups while reducing peak memory by 76.9%–82.0%. At $20\times$–$32\times$ upsampling, the evaluated PUFM† protocol uses three recursive stages, whereas FLUSSO still performs a single five-step stage; the resulting speedups reach $101.6\times$–$147.0\times$ with 89.0%–92.8% lower peak memory. These speedups are specific to the evaluated recursive PUFM† protocol. FLUSSO instead changes candidate and output counts within one stage (Appendix D.1). Its latency drops at $12\times$ because all candidates are retained and global FPS is omitted. Consequently, profiled NN GFLOPs do not directly predict end-to-end runtime. At $4\times$ upsampling, for example, FLUSSO has a higher profiled cost (1032.3 vs. 361.1 GFLOPs), yet achieves both lower latency and lower peak memory. Further stage-wise runtime analysis and comparisons are provided in Appendix D.2 and Appendix D.3, respectively.

Generalization to unseen upsampling ratios. We evaluate a single $4\times$-trained checkpoint on PU1K at seven unseen ratios from $5\times$ to $32\times$. All ratios use the same learned parameters and one five-step generative stage, with global FPS applied only when the candidate count exceeds the requested output size. Competing methods are rerun using their official pretrained checkpoints on the same inputs, with matched output cardinalities and a shared implementation of CD, HD, and P2F. As shown in Table 3, FLUSSO achieves the lowest CD, HD, and P2F across all 21 evaluated ratio–metric combinations. At $32\times$ upsampling, it obtains CD/HD/P2F of $0.147/0.539/1.604$, compared with the corresponding second-best values of $0.180/0.809/2.313$, yielding reductions of 18.3%, 33.4%, and 30.7%, respectively. These results support generalization to unseen output cardinalities without

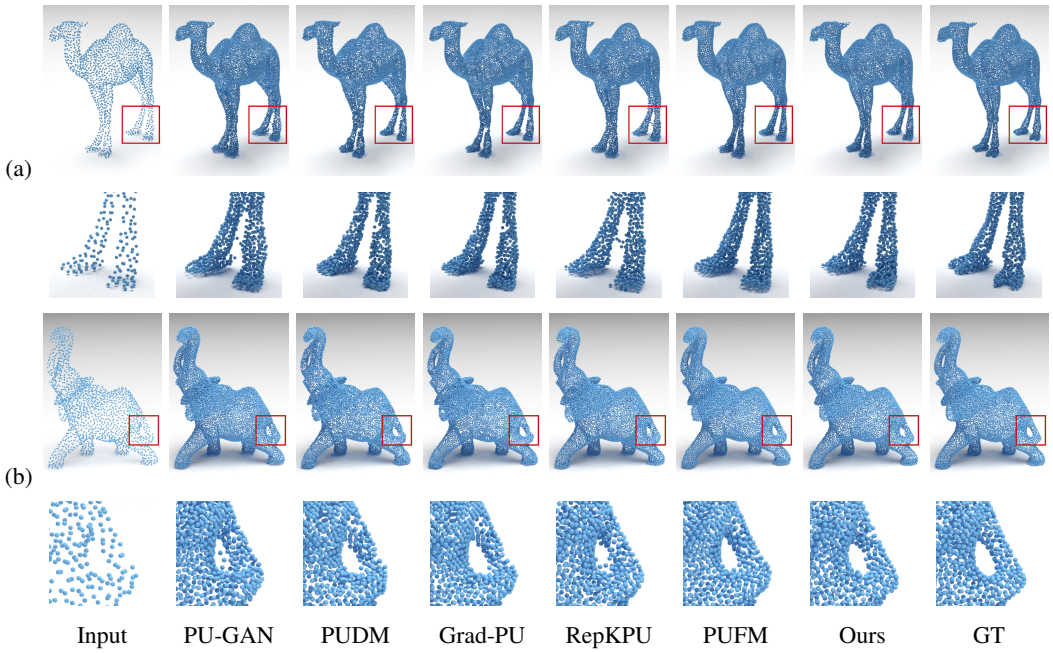

Figure 2: Qualitative comparison of point cloud upsampling methods on (a) camel and (b) elephant. For each object, red boxes mark corresponding regions, with enlarged views directly below. The close-ups show the camel's rear legs and the elephant's tail.

Table 3: Generalization to unseen upsampling ratios on PU1K. All models are evaluated on the same 2,048-point inputs and target cardinalities. Our model is trained only at $4\times$ and evaluated at the listed unseen ratios without retraining. † denotes the rerun PUFM checkpoint. CD/HD/P2F are reported in units of $10^{-4}/10^{-3}/10^{-3}$. Best and second-best results are shown in bold and underline.

	Grad-PU			PUDM			PUFM†			Ours		
Ratio	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓
$5\times$	0.530	<u>0.669</u>	<u>1.955</u>	0.662	0.707	3.293	<u>0.475</u>	0.847	2.108	**0.432**	**0.466**	**1.668**
$7\times$	0.465	0.844	2.493	0.543	<u>0.707</u>	2.944	<u>0.376</u>	0.825	<u>2.103</u>	**0.341**	**0.470**	**1.647**
$8\times$	0.457	0.960	2.768	0.551	<u>0.743</u>	3.238	<u>0.343</u>	0.829	<u>2.102</u>	**0.313**	**0.454**	**1.620**
$12\times$	0.478	1.453	3.669	0.485	<u>0.744</u>	3.202	<u>0.263</u>	0.834	<u>2.042</u>	**0.250**	**0.451**	**1.419**
$20\times$	0.593	2.548	4.974	0.400	<u>0.770</u>	3.384	<u>0.213</u>	0.859	<u>2.284</u>	**0.183**	**0.463**	**1.532**
$24\times$	0.664	3.057	5.533	0.366	<u>0.763</u>	3.197	<u>0.198</u>	0.804	<u>2.291</u>	**0.167**	**0.470**	**1.566**
$32\times$	0.833	4.230	6.565	0.354	<u>0.809</u>	3.289	<u>0.180</u>	0.941	<u>2.313</u>	**0.147**	**0.539**	**1.604**

fine-tuning or recursive model application. At $5\times$–$12\times$, the local particle count remains 24 and only output selection changes; ratios above $12\times$ require larger particle sets. Appendix C additionally tests smaller unseen particle counts without output selection.

Robustness and qualitative real-world transfer. We evaluate robustness to input noise and zero-shot transfer from synthetic training to real sensor data. For each input point in the original evaluation coordinates, we use $x_i^{\text{phys,noise}} = x_i^{\text{phys}} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \eta^2 I_3)$ and η is the per-coordinate standard deviation in that frame; I_3 is the 3×3 identity matrix. The noisy input is then globally normalized using the standard preprocessing pipeline, while the ground-truth point cloud and reference mesh remain clean; the noise is not clipped. As shown in Table 4, FLUSSO achieves the lowest CD and HD across both noise levels, yielding CD/HD of $1.707/1.944$ at $\eta = 0.01$ (vs. $1.798/2.176$ for Grad-PU) and retaining the lowest CD/HD at $\eta = 0.02$ ($3.498/5.586$). Grad-PU achieves lower P2F, so the gains in point-set distances do not extend to every metric. We further apply our model directly to real-world KITTI (Geiger et al., 2013) and ScanNet (Dai et al., 2017) scans featuring irregular sampling and sensor noise. Given the absence of dense ground truth, we

Table 4: Robustness to noisy inputs on PUGAN. Results are reported at $4\times$. CD / HD / P2F are reported in units of $10^{-4}/10^{-3}/10^{-3}$. Best results are in bold.

Method	$\eta = 0.01$			$\eta = 0.02$		
	CD ↓	HD ↓	P2F ↓	CD ↓	HD ↓	P2F ↓
RepKPU	1.712	4.052	7.100	4.055	8.775	16.491
Grad-PU	1.798	2.176	**6.610**	3.598	5.613	**13.138**
PUFM	1.785	2.448	7.790	4.074	6.958	17.090
FLUSSO	**1.707**	**1.944**	7.241	**3.498**	**5.586**	14.695

evaluate qualitatively against PUFM[†]. As shown in Figure 3, FLUSSO better recovers fine local geometry (e.g., the KITTI bicyclist) while preserving LiDAR scan-line structure, and produces more uniform samples on ScanNet with fewer visible gaps, providing qualitative evidence that the learned local generation dynamics transfer to both outdoor and indoor environments.

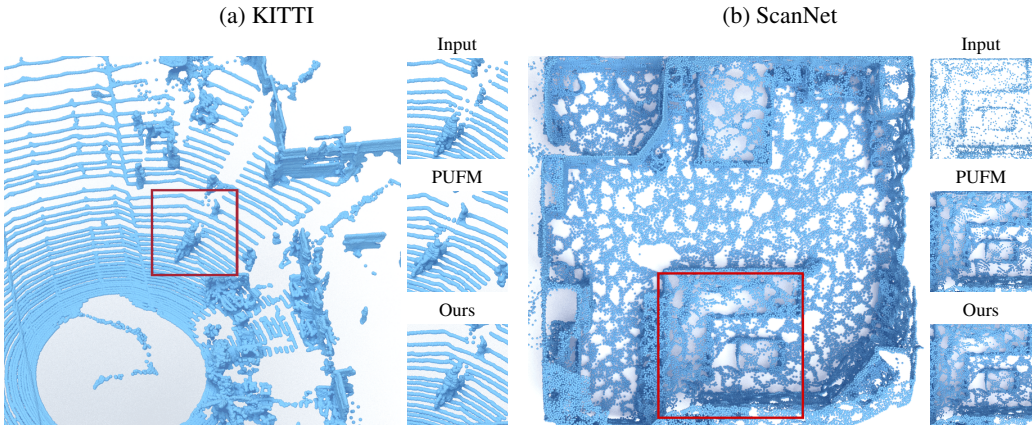

Figure 3: Qualitative comparison on real-world scans from (a) KITTI and (b) ScanNet. Each large view shows the output of our method. Red boxes indicate the regions enlarged in the corresponding right-hand column, which compares Input, PUFM, and Ours from top to bottom. Identical crop coordinates are used across methods within each scene.

4.3 Ablation Studies

Table 5 compares training objectives and coupling selection. All variants share the same velocity architecture, training data, and optimization settings, differing only in the training objective or coupling construction. The single-objective variants test whether conditional velocity targets and model-induced endpoint supervision provide complementary signals. We further replace the proposed soft-weighted top-K coupling with a single minimum-cost sliced plan while keeping the number of randomized quasi-Monte Carlo (RQMC) projection directions unchanged. These variants compare training components under the same optimization schedule; top-K uses more velocity evaluations than top-1 and therefore does not have matched training compute. On PUGAN, the full joint model achieves the lowest CD, HD, and P2F (1.055, 0.919, and 1.731), improving on both single-objective variants. Relative to the joint top-1 variant, soft top-K reduces HD from 1.034 to 0.919 and P2F from 1.782 to 1.731. These results support the combined objective, but do not separately measure accumulated trajectory error.

Table 5: Training-objective and coupling ablations on PUGAN at $4\times$. CD/HD/P2F are reported in units of $10^{-4}/10^{-3}/10^{-3}$.

Objective	Coupling	CD ↓	HD ↓	P2F ↓
SOT-FM only	Soft top-K	1.116	1.135	2.174
Endpoint only	N/A	1.064	0.953	1.893
Joint	Min-cost top-1	1.064	1.034	1.782
Joint (full)	Soft top-K	**1.055**	**0.919**	**1.731**

5 Conclusions

We presented FLUSSO, a point cloud upsampling framework that combines sparse global context with shared local flows. By encoding global geometry once and confining particle interactions to anchor-centered supports, the framework avoids repeated global processing of the dense candidate cloud. SOT-FM provides transport-informed velocity supervision, while multi-step endpoint matching constrains the terminal geometry reached by model-induced rollouts. Experiments on PU1K and PUGAN demonstrate strong reconstruction quality and favorable latency and memory efficiency under the evaluated protocols. The same $4\times$-trained model supports the tested unseen $5\times$–$32\times$ ratios without retraining or recursive model application. Together, these results support locally generative, globally conditioned flows as a practical design for efficient, ratio-flexible point cloud upsampling.

Limitations and future work. Generation cost increases with the local particle count at high output ratios. Future work includes geometry-adaptive particle allocation, more efficient candidate selection, and extending the qualitative real-sensor evaluation to quantitative benchmarks.

AI USE STATEMENT

Generative AI assisted with English-language editing, notation and consistency checks, literature and reference verification, and interpretation of the reported comparisons. The authors take responsibility for the accuracy, originality, and integrity of the manuscript.

REPRODUCIBILITY STATEMENT

Section 3 describes the local flow construction, sliced optimal-transport flow matching, multi-step endpoint distribution matching, and ratio-flexible generation. Appendices A and B detail the network architecture, optimization settings, loss weights, sampling procedures, and ratio-specific inference configurations. Section 4.1 specifies the datasets, benchmark splits, input construction, evaluation metrics, and baseline protocols, distinguishing published scores from our evaluation of the released PUFM checkpoint. All output ratios share a single checkpoint trained at $4\times$ and a fixed five-step Euler integration schedule. Appendix C examines particle-count generalization and candidate selection through controlled inference ablations.

REFERENCES

Laetitia Chapel, Romain Tavenard, and Samuel Vaiter. Differentiable generalized sliced wasserstein plans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 26, 2013.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5828–5839, 2017.

Yi Du, Zhipeng Zhao, Shaoshu Su, Sharath Golluri, Haoze Zheng, Runmao Yao, and Chen Wang. SuperPC: A single diffusion model for point cloud completion, upsampling, denoising, and colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16953–16964, 2025.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Doron Haviv, Aram-Alexandre Pooladian, Dana Pe'er, and Brandon Amos. Wasserstein flow matching: Generative modeling over families of distributions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22238–22258, 2025. URL https://proceedings.mlr.press/v267/haviv25a.html.

Yun He, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Grad-PU: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5354–5363, 2023.

Ka-Hei Hui, Chao Liu, Xiaohui Zeng, Chi-Wing Fu, and Arash Vahdat. Not-so-optimal transport flows for 3D point cloud generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: A point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7203–7212, 2019.

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 (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xinran Liu, Rocio Diaz Martin, Yikun Bai, Ashkan Shahbazi, Matthew Thorpe, Akram Aldroubi, and Soheil Kolouri. Expected sliced transport plans. In *International Conference on Learning Representations (ICLR)*, 2025a.

Yanzhe Liu, Rong Chen, Yushi Li, Yixi Li, and Xuehou Tan. SPU-PMD: Self-supervised point cloud upsampling via progressive mesh deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5188–5197, 2024.

Zhi-Song Liu, Zijia Wang, and Zhen Jia. Arbitrary point cloud upsampling via dual back-projection network. In *IEEE International Conference on Image Processing (ICIP)*, pp. 1470–1474, 2023.

Zhi-Song Liu, Chenhang He, Roland Maier, and Andreas Rupp. PUFM++: Point cloud upsampling via enhanced flow matching. *arXiv preprint arXiv:2512.20988*, 2025b. URL https://arxiv.org/abs/2512.20988.

Zhi-Song Liu, Chenhang He, Yakun Ju, and Lei Li. Pufm: Efficient point cloud upsampling via flow matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 7458–7466, 2026.

Luqing Luo, Lulu Tang, Wanyi Zhou, Shizheng Wang, and Zhi-Xin Yang. PU-EVA: An edge-vector based approximation solution for flexible-scale point cloud upsampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16208–16217, 2021.

Guillaume Mahey, Laetitia Chapel, Gilles Gasso, Clément Bonet, and Nicolas Courty. Fast optimal transport through sliced Wasserstein generalized geodesics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Aihua Mao, Zihui Du, Junhui Hou, Yaqi Duan, Yong-Jin Liu, and Ying He. PU-Flow: A point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4964–4977, 2023.

Khai Nguyen, Nicola Bariletto, and Nhat Ho. Quasi-Monte Carlo for 3D sliced Wasserstein. In *International Conference on Learning Representations (ICLR)*, 2024.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28100–28127, 2023. URL https://proceedings.mlr.press/v202/pooladian23a.html.

Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. PU-GCN: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11683–11692, 2021.

Yue Qian, Junhui Hou, Sam Kwong, and Ying He. PUGeo-Net: A geometry-centric network for 3D point cloud upsampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 752–769, 2020.

Shi Qiu, Saeed Anwar, and Nick Barnes. PU-Transformer: Point cloud upsampling transformer. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 2475–2493, 2022.

Wentao Qu, Yuantian Shao, Lingwu Meng, Xiaoshui Huang, and Liang Xiao. A conditional denoising diffusion probabilistic model for point cloud upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20786–20795, 2024.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, pp. 435–446, 2011.

Yi Rong, Haoran Zhou, Kang Xia, Cheng Mei, Jiahao Wang, and Tong Lu. RepKPU: Point cloud upsampling with kernel point representation and deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21050–21060, 2024.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

Zijian Xiao, Yining Xu, Yingjie Huang, and Li Yao. PUNO: A neural operator framework for point cloud upsampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11042–11050, 2026.

Shuquan Ye, Dongdong Chen, Songfang Han, Ziyu Wan, and Jing Liao. Meta-PU: An arbitrary-scale upsampling network for point cloud. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3206–3218, 2022.

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2790–2799, 2018.

Boqian Zhang, Shen Yang, Hao Chen, Chao Yang, Jing Jia, and Guang Jiang. Point cloud upsampling using conditional diffusion module with adaptive noise suppression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16987–16996, 2025.

Wenbo Zhao, Xianming Liu, Zhiwei Zhong, Junjun Jiang, Wei Gao, Ge Li, and Xiangyang Ji. Self-supervised arbitrary-scale point clouds upsampling via implicit neural representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1999–2007, 2022.

Supplementary Material for "Efficient Point Cloud Upsampling via Local Flow Matching"

This supplement provides the network architecture (Appendix A), training and inference settings (Appendix B), controlled inference ablations (Appendix C), computational and runtime analyses (Appendix D), and additional qualitative comparisons (Appendix E). We use the notation of Sec. 3: N is the sparse input cardinality, G the number of anchors, q the particles per anchor, $R = qG$ the candidate cardinality, and $M = rN$ the requested output cardinality. In particular, the output ratio r and local particle count q are distinct quantities.

A Network Architecture

Figure 4 expands the velocity-field architecture in Sec. 3.3. The network separates sparse global context encoding from repeated local particle updates.

Sparse context encoder. A 16-neighbor EdgeConv encoder extracts local geometric features, followed by four self-attention blocks with eight heads to produce $H \in \mathbb{R}^{N\times C}$, where $C = 256$. Global attention operates on the sparse input, and these context features are reused throughout the local flow integration. For anchor $a_j = x_{\iota_j}$, the anchor feature h_{ι_j} provides the global conditioning signal, while the features associated with $\mathcal{B}_j^X$ provide spatially local context.

Local velocity network. The anchor-relative particle coordinates $Z_j \in \mathbb{R}^{q\times 3}$ are embedded as tokens and processed by four conditioned transformer (DiT) blocks, each with four attention heads. Each block applies self-attention among the anchor's q particles, cross-attention to sparse-support features, and a per-token multilayer perceptron (MLP). The anchor feature and flow time form the conditioning signal. Each block projects this signal into nine d-dimensional vectors, where $d = 256$ is the local token width: one shift, scale, and residual gate for each of the self-attention, cross-attention, and MLP sublayers. The final conditioned normalization and shared linear head map the tokens to $v_\theta(Z_j, t, c_j) \in \mathbb{R}^{q\times 3}$.

Parameter sharing and variable cardinality. All anchors use the same encoder and velocity-field parameters. The particle attention and per-token velocity head accept different values of q, so changing the output ratio requires no ratio-specific prediction layer. The total parameter count remains 10.4 million. Depending on the requested ratio, inference changes the particle count, the number of candidates retained by global FPS, or both, as specified in Appendix B.2.

B Implementation Details

Table 6 collects the settings used in the main experiments. A single checkpoint trained at $4\times$ is used for all reported output ratios; the inference ablation in Appendix C changes only candidate generation and output selection.

B.1 Training Configuration

Data and supports. We use the combined PU1K–PUGAN training data and benchmark splits specified in Sec. 4.1. Each dense patch contains 1,024 points sampled from the reference mesh by Poisson-disk sampling. Its 256-point sparse input is obtained by FPS with a randomly initialized first point. For the even input sizes used in our experiments, FPS selects $G = N/2$ anchors. Each anchor uses radius $s_j = d_5(a_j, X)$, measured to the fifth nearest non-self sparse point ($K_b = 5$, $\beta = 1$). The same radius defines the sparse conditioning and dense supervision supports.

Source and target particles. In the globally normalized input frame, training samples $q_{\text{train}} = 24$ source particles uniformly from each anchor-centered ball. Local targets are obtained by FPS within the corresponding dense support, repeating the FPS ordering cyclically when fewer than 24 points are available. Both sets are represented relative to the anchor without division by s_j. The $4\times$ training

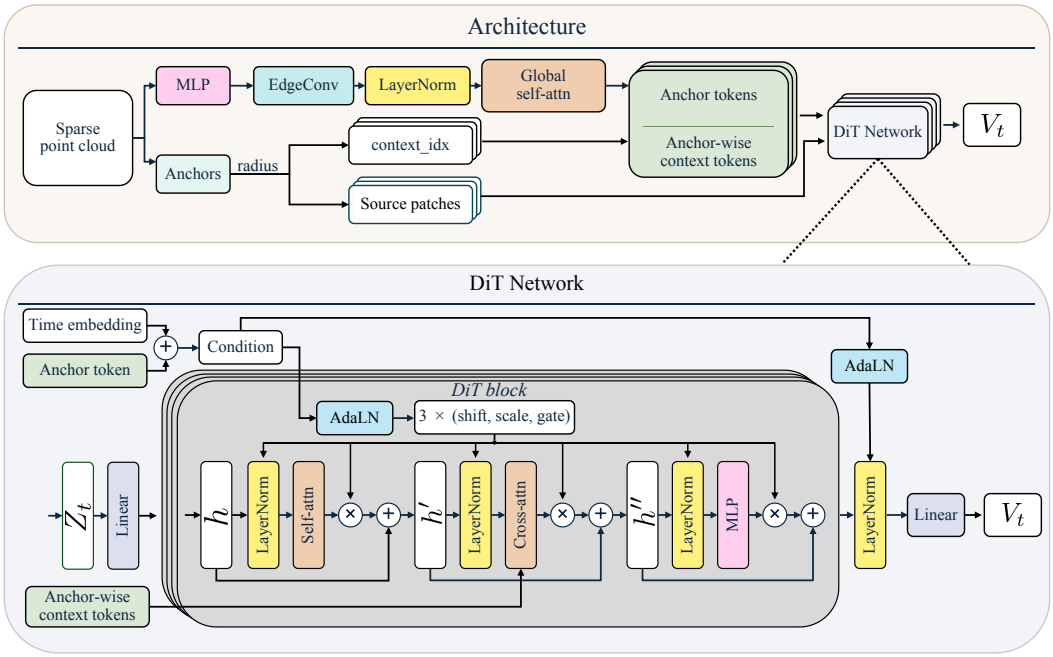

Figure 4: **Network architecture of FLUSSO.** The sparse encoder extracts globally contextualized features. For each anchor, the anchor feature and sparse-support features condition a shared local DiT that evolves the source particles and predicts one 3D velocity per particle. In the diagram, Z_t is the current local coordinate matrix and $V_t = v_\theta(Z_t, t, c_j)$ its predicted velocity; h, h', h'' denote intermediate local token matrices, distinct from the sparse features h_i. The label `context_idx` denotes sparse-support indices.

ratio describes the 256-to-1,024 data pair; q_{train} independently specifies the size of each local source and target set used for supervision.

Couplings and endpoint supervision. SOT-FM uses $L_f = 128$ low-discrepancy projection directions, with an independent Haar rotation for each training shape and a common direction set across its anchors. Each direction induces a sorted 1D matching, which is evaluated using the original 3D coordinates. We normalize the matching costs by $\max(s_j^2, \epsilon_s)$ as in Sec. 3.4, retain the $K = 4$ lowest-cost plans, and use temperature $T_c = 0.05$ for their soft weights. The individual plans define separate conditional flow-matching paths. One sampled flow time is shared across the anchors and retained plans of a training shape. LEDM starts from the same source particles and differentiates through $K_e = 5$ Euler steps, using an independently randomized set of $L_e = 128$ directions to match the terminal local distributions. The joint loss weights are $\lambda_{\text{fm}} = 1$ and $\lambda_{\text{end}} = 7$.

Optimization. We train for 200 epochs using Adam without weight decay, with a cosine learning-rate schedule from 10^{-4} to 10^{-6}. The global batch size is 64, distributed as 32 samples on each of two GPUs, without gradient accumulation. Gradients are clipped to a maximum norm of 5.0.

B.2 Ratio-Specific Inference Configuration

All output ratios use the same learned parameters, fixed Sobol source initialization, and a single five-step Euler integration over $[0, 1]$. Dense targets, SOT-FM coupling construction, and the endpoint loss are used only during training. For the default configuration at integer ratios r and even input sizes N, we set

$$q_{\text{test}} = \max(24, 2r), \qquad R = Gq_{\text{test}}, \qquad M = rN, \tag{6}$$

where R and M denote the candidate and requested output cardinalities, respectively. Since $G = N/2$, ratios below $12\times$ use $q_{\text{test}} = 24$ and generate the same over-complete candidate cloud with $R = 12N$, from which global FPS selects $M = rN$ points. At and above $12\times$, $R = M$, so all generated candidates are retained and the final FPS operation is omitted. Table 7 gives the resulting cardinalities

Table 6: Implementation settings of FLUSSO. N is the input size and r the output ratio.

Component	Configuration
Training data	256 input points; 1,024 target points; $4\times$ training
Sparse encoder	$C = 256$; 16-NN EdgeConv; 4 attention blocks; 8 heads
Local velocity field	$d = 256$; 4 DiT blocks; 4 attention heads
Anchors/support	FPS anchors; $G = N/2$; $K_b = 5$; $\beta = 1$
Source initialization	Random uniform ball samples for training; fixed Sobol samples for inference
Default particles	$q_{\text{train}} = 24$; $q_{\text{test}} = \max(24, 2r)$
SOT-FM	$L_f = 128$; top-$K = 4$ couplings; $T_c = 0.05$
LEDM	$L_e = 128$; $K_e = 5$ rollout steps
Loss weights	$\lambda_{\text{fm}} = 1$; $\lambda_{\text{end}} = 7$
Optimization	Adam; 200 epochs; LR $10^{-4} \to 10^{-6}$ with cosine decay; weight decay 0
Batch	2 GPUs; 32 samples/GPU; global batch size 64
Gradient clipping	Maximum gradient norm 5.0
Inference	$S = 5$ Euler steps; global FPS only when $R > rN$

Table 7: Default ratio-specific inference configuration of FLUSSO for $N = 2{,}048$ input points and $G = 1{,}024$ anchors. All ratios use the same $4\times$-trained checkpoint and five Euler steps. R and M denote the candidate and output cardinalities, respectively.

Ratio	q_{test}	Candidates R	Outputs M	Global FPS
$4\times$	24	24,576	8,192	Yes
$5\times$	24	24,576	10,240	Yes
$7\times$	24	24,576	14,336	Yes
$8\times$	24	24,576	16,384	Yes
$12\times$	24	24,576	24,576	No
$16\times$	32	32,768	32,768	No
$20\times$	40	40,960	40,960	No
$24\times$	48	49,152	49,152	No
$32\times$	64	65,536	65,536	No

for the 2,048-point test inputs. The encoder, learned velocity field, and integration schedule remain fixed throughout. At $4\times$–$12\times$, $q_{\text{test}} = q_{\text{train}}$; these configurations change the final output cardinality without changing the local particle count. Above $12\times$, local particle sets exceed the training size.

B.3 Evaluation Metrics

We report Chamfer distance (CD), squared Hausdorff distance (HD), and point-to-surface distance (P2F). For indexed predicted and reference point collections P and Q, counting repeated entries, define

$$d_{P\to Q}(p) = \min_{p'\in Q} \|p - p'\|_2^2,$$

$$\mathrm{CD}(P,Q) = \frac{1}{2}\left(\frac{1}{|P|}\sum_{p\in P} d_{P\to Q}(p) + \frac{1}{|Q|}\sum_{p'\in Q} d_{Q\to P}(p')\right),$$

$$\mathrm{HD}(P,Q) = \max\left\{\max_{p\in P} d_{P\to Q}(p), \max_{p'\in Q} d_{Q\to P}(p')\right\}.$$

Here $d_{Q\to P}$ is defined analogously. CD averages the two directional means, whereas HD takes the largest squared deviation. For reference mesh surface $\mathcal{S}$, P2F is $|P|^{-1}\sum_{p\in P}\inf_{s\in\mathcal{S}}\|p - s\|_2$. It measures surface proximity rather than coverage. Predictions and references are compared in the original coordinates; CD and HD use squared distance units, whereas P2F uses distance units.

Table 8: Ablation of candidate generation and output selection at unseen upsampling ratios. All variants use the same checkpoint trained exclusively at $4\times$. Random results are reported as mean $\pm$ standard deviation over five selection seeds. Values in parentheses after q_{test} denote the candidate-to-output ratio R/M. CD is multiplied by 10^4, whereas HD and P2F are multiplied by 10^3. Latency is the mean per-shape inference time over the 127 PU1K test shapes on the same isolated NVIDIA RTX 4090 used for the efficiency experiments. Lower values are better.

Ratio	Selection	q_{test} (R/M)	CD ↓	HD ↓	P2F ↓	Latency (ms) ↓
	FPS	24 (2.40)	**0.4317**	**0.4658**	1.6680	159.7
$5\times$	Random	24 (2.40)	0.5121 ± 0.0004	0.6091 ± 0.0060	$\mathbf{1.4159\pm0.0017}$	105.1 ± 0.3
	Direct	10 (1.00)	0.4954	0.5120	1.4213	**69.7**
	FPS	24 (1.71)	**0.3414**	**0.4702**	1.6467	182.5
$7\times$	Random	24 (1.71)	0.3844 ± 0.0002	0.5227 ± 0.0090	1.4179 ± 0.0012	105.2 ± 0.5
	Direct	14 (1.00)	0.3755	0.4870	**1.3914**	**77.2**
	FPS	24 (1.50)	**0.3126**	**0.4539**	1.6200	193.9
$8\times$	Random	24 (1.50)	0.3437 ± 0.0001	0.4957 ± 0.0092	1.4169 ± 0.0007	105.5 ± 0.4
	Direct	16 (1.00)	0.3378	0.4591	**1.3851**	**81.2**

C Particle-Count Generalization and Output Selection

The default $5\times$, $7\times$, and $8\times$ results in Table 3 use $q_{\text{test}} = 24$, the same local particle count as training, followed by global FPS. They establish flexibility in the final output cardinality but do not by themselves test generalization to unseen local particle counts. We therefore first compare three inference strategies on 2,048-point PU1K test inputs at $5\times$, $7\times$, and $8\times$, using the same $4\times$-trained checkpoint and five-step integration schedule without fine-tuning. We then evaluate the whole-shape uniformity of these strategies on 27 PUGAN test shapes at $4\times$, $5\times$, $7\times$, and $8\times$.

Controlled variants. We evaluate FPS, Random, and Direct on PU1K at $r \in \{5, 7, 8\}$ and PUGAN at $r \in \{4, 5, 7, 8\}$, with $M = rN$ outputs. With $N/2$ anchors, FPS and Random share $q_{\text{test}} = 24$ and $R = 12N$ candidates, selecting M by global FPS and uniform sampling without replacement, respectively. Direct uses $q_{\text{test}} = 2r$, below the training count of 24, and retains all $R = M$ candidates. Thus, FPS versus Random isolates the selection rule, whereas Direct changes both candidate count and selection.

Whole-shape uniformity protocol. For PUGAN, we sample 1,000 disk centers and $n_{\text{surf}} = 65{,}536$ independent surface points from each complete reference mesh. We set the disk radius to $\rho = \alpha \,\text{median}_j(s_j)$ for $\alpha \in \{0.5, 1, 2\}$, using anchor support radii s_j from the default $q_{\text{test}} = 24$ run. We measure uniformity using the normalized count variance, denoted NUC. For disk i, let C_i and k_i count output points and reference surface samples, respectively. Its expected output count is $E_i = Mk_i/n_{\text{surf}}$, and

$$\text{NUC} = \text{Var}_{i:E_i>0}(C_i/E_i).$$

Lower NUC indicates more uniform relative occupancy across the sampled disks. Generated points are projected onto the mesh only for this uniformity evaluation; the CD, HD, and P2F evaluations use the original predictions. All variants and cardinality-matched ground truth (GT) use identical disks. Table 9 reports the mean of the per-shape NUC values over the 27 PUGAN test shapes.

Effect of candidate selection. In Table 8, FPS yields lower CD and HD than Random at all three ratios. Its relative CD improvement decreases from 15.7% at $5\times$ to 9.0% at $8\times$ as the candidate-to-output ratio falls from 2.40 to 1.50. These results are consistent with FPS improving output coverage when redundant candidates are available. However, Random has lower P2F at every ratio: for example, 1.4159 versus 1.6680 at $5\times$. Thus, selection trades off point-set distances against average surface proximity; FPS does not improve every metric.

Generation with unseen particle counts. Direct produces exactly the requested cardinality and achieves lower CD and HD than Random at every tested ratio. Relative to the rerun baselines in Table 3, Direct has lower HD and P2F at all three ratios. At $5\times$, its CD of 0.4954 is higher than

Table 9: Whole-shape NUC on PUGAN at three disk-radius scales (lower is better). The scale α multiplies the median anchor support radius of each shape. Random is reported as mean $\pm$ standard deviation over five random subsampling repetitions, each averaged over 27 shapes. GT is the cardinality-matched reference point cloud. Bold indicates the best generated variant.

Ratio	Scheme	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$
$4\times$	FPS	**0.088592**	**0.019962**	**0.006056**
	Random	0.263599 ± 0.003350	0.078155 ± 0.000681	0.023657 ± 0.000167
	Direct	0.172238	0.052476	0.017083
	GT	0.073311	0.013361	0.003088
$5\times$	FPS	**0.084202**	**0.020391**	**0.006215**
	Random	0.219439 ± 0.002932	0.067114 ± 0.000785	0.020929 ± 0.000283
	Direct	0.152157	0.048027	0.015694
	GT	0.062530	0.011974	0.002706
$7\times$	FPS	**0.089704**	**0.025292**	**0.008137**
	Random	0.172692 ± 0.000703	0.056986 ± 0.000790	0.019048 ± 0.000313
	Direct	0.135400	0.047578	0.016660
	GT	0.050464	0.009867	0.002283
$8\times$	FPS	**0.094208**	**0.028432**	**0.009181**
	Random	0.155858 ± 0.001014	0.053325 ± 0.000387	0.017409 ± 0.000161
	Direct	0.128592	0.046315	0.015680
	GT	0.048203	0.009544	0.002220

PUFM†'s 0.475 but lower than Grad-PU's 0.530 and PUDM's 0.662. At $8\times$, Direct has lower CD than all three baselines; at $7\times$, it matches PUFM† to three decimal places (0.3755 versus 0.376). Within our variants, FPS retains the best CD and HD, while Direct achieves the lowest P2F at $7\times$ and $8\times$. Direct reduces latency from $159.7/182.5/193.9$ ms to $69.7/77.2/81.2$ ms at $5\times/7\times/8\times$, respectively, corresponding to $2.29\times$–$2.39\times$ speedups.

Whole-shape uniformity on PUGAN. In Table 9, FPS achieves the lowest NUC among the generated variants at every evaluated rate and radius scale. At $\alpha = 0.5$, its reductions relative to Random are 66.4%, 61.6%, 48.1%, and 39.6% at $4\times$, $5\times$, $7\times$, and $8\times$, respectively. Direct also yields lower NUC than Random in all 12 rate–radius settings despite omitting global selection. The ordering FPS $<$ Direct $<$ Random persists across all three radius scales. Direct produces more uniform occupancy than Random, while the FPS–Random comparison isolates the improvement from global selection on identical candidates. Among the generated variants, FPS remains closest to GT in NUC throughout the evaluation.

Together with the default results above $12\times$, the PU1K ablation shows that a single $4\times$-trained checkpoint supports both smaller and larger local particle counts at inference. Direct generation avoids terminal selection and reduces latency, whereas FPS improves CD and HD relative to Random at higher latency and without improving P2F at the tested rates. On PUGAN, Direct yields lower whole-shape NUC than Random without global selection, and FPS achieves the lowest NUC among the generated variants across all evaluated disk radii. These results show that local particle count provides flexible output cardinality, while global selection improves point-set coverage and uniformity when redundant candidates are available.

D Efficiency Analysis

We separate the costs of sparse encoding, local generation, and output selection, then examine measured latency and memory at matched input and output sizes. Flow evaluations and coordinate-gradient updates are counted separately because their computational costs differ.

D.1 Computational Complexity

Let $B = \max_j |\mathcal{B}_j^X|$ be the largest sparse-support size. We fix network depth, head count, and encoder neighborhood size, and use C as a representative feature width. The neural costs below exclude geometric preprocessing and output selection.

Sparse encoding and local integration. Global sparse attention contributes $\mathcal{O}(N^2C)$, and feature projections contribute $\mathcal{O}(NC^2)$. Per-anchor particle self-attention and sparse cross-attention cost $\mathcal{O}((q^2 + qB)C)$ per velocity evaluation. Including projections and MLPs yields

$$\begin{aligned} \mathcal{C}_{\text{enc}} &= \mathcal{O}(N^2C + NC^2), \\ \mathcal{C}_{\text{local}} &= \mathcal{O}\big(G\big[(q^2 + qB)C + (q + B)C^2\big]\big)\,. \end{aligned} \tag{7}$$

The neural portion of inference therefore costs $\mathcal{C}_{\text{enc}} + S\mathcal{C}_{\text{local}}$. For comparison, dense self-attention over all $R = qG$ particles at each step would contribute $\mathcal{O}(Sq^2G^2C)$. Local generation reduces this particular interaction term to $\mathcal{O}(SGq^2C)$, a factor of G, while sparse global conditioning is computed once. This factor applies to the attention term, not the full inference pipeline.

Training-only transport and rollout costs. Constructing L_f sliced couplings requires $\mathcal{O}(GL_f q \log q)$ work for sorting. Scoring their induced 3D correspondences adds $\mathcal{O}(GL_f q)$ work and does not require a dense $q \times q$ transport matrix. The K retained plans require K velocity evaluations, while LEDM adds K_e differentiable rollout steps and $\mathcal{O}(GL_e q \log q)$ work for endpoint projection sorting. The resulting training cost, up to constant factors from backpropagation, is

$$\mathcal{C}_{\text{enc}} + (K + K_e)\mathcal{C}_{\text{local}} + \mathcal{O}(G(L_f + L_e)q \log q)\,. \tag{8}$$

These coupling and loss computations are absent at inference.

Preprocessing and selection. Neighborhood search, anchor FPS, support construction, and source initialization add implementation-dependent costs, measured as Prep. in Table 10. Merging the terminal candidates is linear in R. When $R > M$, conventional iterative global FPS costs $\mathcal{O}(RM)$ distance-update work; when $R = M$, this selection step is omitted. Thus, output selection can dominate changes in runtime even when the neural computation is unchanged. Above $12\times$, $q = 2r$ increases with the requested ratio, so localization does not make generation independent of output cardinality or eliminate the quadratic dependence on q in local self-attention.

D.2 Stage-Wise Runtime Breakdown

The main paper reports end-to-end latency at matched output cardinalities in Table 2. Here we decompose FLUSSO into preparation, sparse encoding, local flow integration, candidate merging, and output selection. Measurements use 2,048-point inputs, FP32 precision, batch size one, and an NVIDIA GeForce RTX 4090 without concurrent GPU workloads. Each timing averages 50 runs after 10 warm-up iterations. Stages are synchronized and timed separately; end-to-end latency is measured independently. The breakdown identifies costs within FLUSSO rather than matching individual stages across architectures.

Fixed particle count through $12\times$. In Table 10, the stage sum differs from the independently measured end-to-end latency by at most 0.74%, relative to the end-to-end measurement. For $4\times$–$12\times$, $q_{\text{test}} = 24$, so the flow cost remains nearly constant at approximately 61 ms. For ratios below $12\times$, however, the candidate cloud is over-complete and Sel./Asm. includes global FPS, whose cost increases as more candidates are retained. At $12\times$, $R = M$, so FPS is omitted and the Sel./Asm. cost drops from 89.26 ms at $8\times$ to 0.03 ms, making $12\times$ inference faster despite its larger output.

Increasing particle count above $12\times$. Above $12\times$, local flow integration becomes the dominant ratio-dependent cost: increasing q_{test} from 32 at $16\times$ to 64 at $32\times$ increases flow latency from 80.68 ms to 170.09 ms, while the remaining components stay nearly constant. The measured growth over these configurations should not be interpreted as an asymptotically linear complexity guarantee: the local attention term remains quadratic in q, as described in Appendix D.1. The latency values in this table and Table 2 come from separate runs; Table 8 instead averages over all 127 PU1K test shapes. Timing comparisons therefore use measurements from the same table.

Table 10: Stage-wise inference latency of FLUSSO. Mean latency in milliseconds over 50 runs after 10 warm-up iterations on an isolated RTX 4090, using 2,048-point inputs, FP32, and batch size one. Prep. includes neighborhood construction, anchor sampling, local-support construction, and source initialization. Sel./Asm. denotes output selection and assembly, including global FPS when $R > M$. Stage sum is computed from separately synchronized component timings, whereas end-to-end latency (E2E) is measured independently over the complete inference pipeline.

Ratio	Stage latency (ms)					Stage sum	E2E	FPS
	Prep.	Encoder	Flow	Merge	Sel./Asm.			
4×	34.27	4.52	61.14	0.01	44.58	144.52	143.86	Yes
5×	34.64	4.52	61.14	0.02	55.95	156.27	155.83	Yes
7×	33.57	4.52	61.20	0.02	77.84	177.15	177.35	Yes
8×	33.31	4.52	61.24	0.02	89.26	188.35	187.83	Yes
12×	33.79	4.51	61.32	0.01	0.03	99.66	98.93	No
16×	34.56	4.51	80.68	0.02	0.03	119.80	120.15	No
20×	34.14	4.52	101.35	0.02	0.04	140.06	139.73	No
24×	33.71	4.52	123.39	0.01	0.03	161.66	161.43	No
32×	34.46	4.53	170.09	0.02	0.04	209.15	209.07	No

Table 11: Efficiency comparison under Grad-PU's released inference protocol. F/G denotes FLUSSO/Grad-PU. Measurements use an isolated RTX 4090, 2,048-point inputs, FP32, and batch size one. Updates report the five Euler evaluations used by FLUSSO and the total number of coordinate-gradient refinement iterations used by Grad-PU. Latency is mean ± standard deviation over 50 runs after 10 warm-ups. Speedup is $t_{\mathrm{G}}/t_{\mathrm{F}}$, where t_{G} and t_{F} are the latencies of Grad-PU and FLUSSO, respectively; memory is peak GPU allocation.

Ratio	Stages F/G	Updates F/G	Params F/G (M)	Latency (ms)		Speedup	Memory F/G (MiB)
				FLUSSO	Grad-PU		
5×	1/1	5/10	10.4/0.067	156.0 ± 0.6	152.9 ± 0.2	0.98×	371.3/337.2
7×	1/1	5/10	10.4/0.067	177.9 ± 0.8	269.3 ± 0.3	1.51×	371.3/472.0
8×	1/1	5/10	10.4/0.067	189.8 ± 0.8	336.9 ± 0.2	1.78×	371.3/539.3
12×	1/1	5/10	10.4/0.067	100.7 ± 0.2	689.6 ± 0.4	6.85×	371.3/810.2
16×	1/2	5/20	10.4/0.067	119.4 ± 0.6	1263.8 ± 0.5	10.58×	475.4/1078.4
20×	1/1	5/10	10.4/0.067	140.0 ± 0.6	1761.5 ± 2.9	12.58×	579.5/1348.6
24×	1/1	5/10	10.4/0.067	162.1 ± 0.3	2482.8 ± 0.5	15.32×	683.6/1617.1
32×	1/1	5/10	10.4/0.067	208.6 ± 0.4	4277.2 ± 0.7	20.50×	891.8/2155.9

D.3 Efficiency Comparison with Grad-PU

To complement the PUFM† comparison in the main paper, we benchmark Grad-PU under its released inference protocol. Both methods use 2,048-point inputs, FP32 precision, batch size one, and the same isolated RTX 4090. Grad-PU uses one refinement stage at each evaluated ratio except $16\times$, where its released implementation applies the $4\times$ model twice. FLUSSO always uses one five-step stage. Table 11 counts Euler evaluations for FLUSSO and coordinate-gradient updates for Grad-PU.

Latency and memory tradeoffs. At $5\times$, Grad-PU is marginally faster and uses less peak memory than FLUSSO (152.9 versus 156.0 ms and 337.2 versus 371.3 MiB). At every tested ratio from $7\times$ onward, FLUSSO is faster and uses less peak memory. Speedup grows from $1.51\times$ at $7\times$ to $20.50\times$ at $32\times$, where peak memory is 58.6% lower. The $16\times$ comparison includes Grad-PU's two recursive stages, as specified above. Grad-PU has substantially fewer parameters (0.067 million versus 10.4 million), so parameter count alone does not explain the measured runtime and memory differences.

Grad-PU requires backward passes through its distance network, custom PointOps kernels, and CPU neighborhood extraction. These operations are not fully covered by the NN FLOP profiler, so we compare end-to-end latency and peak allocated memory.

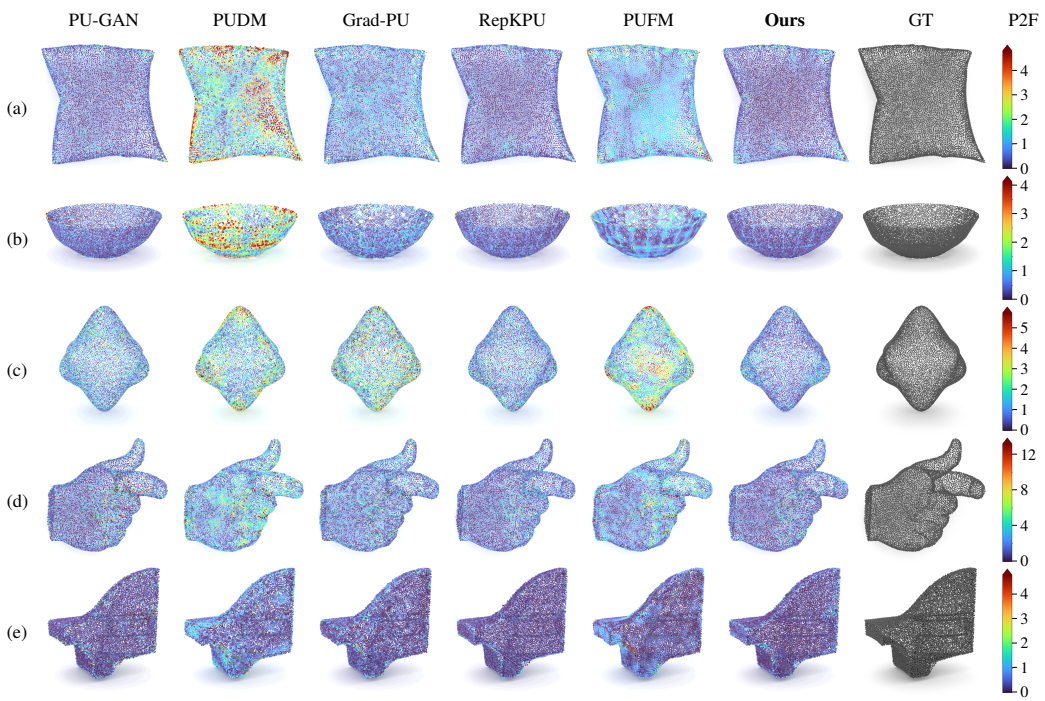

Figure 5: **P2F visualization on PU1K under $4\times$ upsampling.** Colors encode point-to-surface (P2F) error, with cooler colors indicating smaller errors and warmer colors indicating larger errors. Colorbar values are in units of 10^{-3}; color scales vary across rows. Ground truth is gray.

E Additional Qualitative Results

These visualizations complement Sec. 4.2 by showing spatial surface errors and local point coverage.

E.1 Visualization of Point-to-Surface Errors

Protocol and interpretation. We rerun the compared methods using their officially released pretrained weights under the PU1K $4\times$ setting. Figure 5 compares PU-GAN, PUDM, Grad-PU, RepKPU, PUFM, and FLUSSO, with ground-truth point clouds in gray. Each predicted point is colored by its unsigned Euclidean distance to the corresponding reference surface. Cooler colors indicate smaller errors; colorbar values are in units of 10^{-3}. All methods within a row use the same color scale, but the scale varies across shapes. Colors should therefore be compared within each row rather than interpreted as a common absolute error across rows.

Observations. The selected examples show how errors are distributed over curved surfaces, boundaries, and thin structures. FLUSSO produces predominantly low-error surfaces in these views, while several competing predictions exhibit more concentrated high-error regions. The gray reference shows the target geometry. These selected examples illustrate local behavior; dataset-level conclusions follow the quantitative results in Table 1.

E.2 Local Geometry and Enlarged Details

Figure 6 supplements Figure 2 with a chair example. The upper row shows the sparse input, competing predictions, FLUSSO, and the dense reference. Red boxes identify the regions shown at higher magnification in the lower row. The close-ups reveal thin-structure continuity, local coverage, and displaced points, complementing the P2F views in Figure 5.

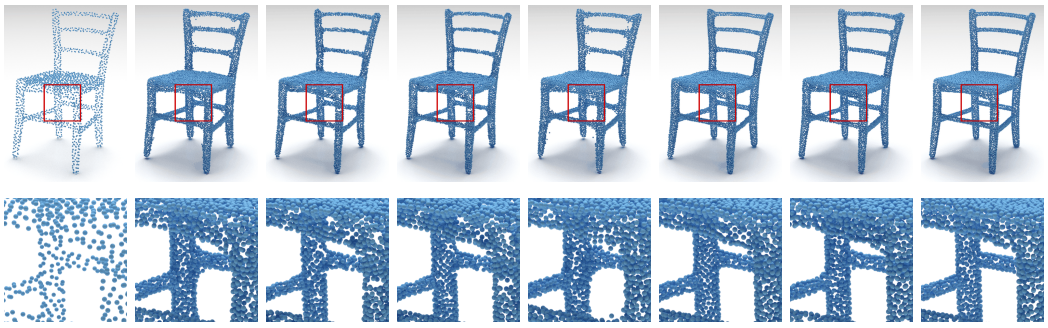

Figure 6: Qualitative comparison of point cloud upsampling methods on a chair. Red boxes mark the regions enlarged in the second row. GT denotes ground truth.