VISFLOW: INTERPRETABLE FLOW MATCHING FOR VISIBILITY RECONSTRUCTION, CHARACTERIZATION, AND RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Methods for learning continuous representations, such as latent manifolds, are fundamental and ubiquitous in modern machine learning. Yet many geometric objects combine continuous shape information with discrete combinatorial structure. This combination presents a fundamental challenge for representation learning because small geometric perturbations can induce abrupt changes in the underlying discrete representation. As a result, the smoothness assumptions that make continuous representations effective no longer hold. Polygons provide a canonical example. Their vertex locations vary continuously, but infinitesimal changes in geometry can dramatically alter the associated visibility graph. Existing methods address the inverse mapping from visibility graphs to polygon geometry using vertex-, triangulation-, or signed-distance-function (SDF)-based representations. Vertex- and triangulation-based approaches construct polygons through explicit geometric operations. However, they are less effective at modeling the relationship between geometry and visibility. SDF-based approaches capture this relationship more naturally. Their intermediate states, however, do not correspond to explicit stages of polygon construction.

We present **VisFlow**, a novel, interpretable flow formulation that combines continuous SDF representations with explicit polygon evolution. Our method models polygon generation as rectified flow in the latent space of continuous SDFs. Standard rectified flow typically constructs linear paths between samples from an unstructured source distribution and the target distribution. The resulting intermediate states are not tied to stages of polygon formation. In contrast, **VisFlow** learns approximately linear latent paths to represent explicit stages of polygon growth along SDF normals. We evaluate **VisFlow** on *Visibility Reconstruction* and *Visibility Characterization*. *Visibility Reconstruction* generates one polygon consistent with a visibility graph G, while *Visibility Characterization* characterizes all such polygons. **VisFlow** maintains **20%** more visibility relations on *Visibility Reconstruction* and improves coverage by **30%** on *Visibility Characterization* compared to state-of-the-art methods. We further show that **VisFlow** generalizes to *Visibility Recognition*, where the input graph is not guaranteed to be valid. We also demonstrate that **VisFlow** extends to a real-world protein reconstruction problem by applying its trajectory-supervision formulation to generate protein structures from contact graphs.

1 INTRODUCTION

Learning the mapping from a graph to its geometric realization arises in problems such as molecular conformation generation, spatial layout reconstruction, and mesh generation. Existing approaches often use continuous geometric attributes, such as coordinates or distances, alongside graph connectivity to represent the underlying geometry. Local perturbations produce small changes in these attributes, keeping nearby geometric realizations close in the representation space. The resulting continuity forms a smooth geometric manifold that learning models can exploit. Purely discrete graph representations lack an analogous continuous structure. Small geometric perturbations can change several edge relations simultaneously, causing similar realizations to map to substantially

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.

different graph representations. Therefore, learning the inverse mapping from a discrete graph to its geometric realization is more difficult because nearby geometries are not necessarily nearby in the representation space.

In this paper, we study one such discrete representation: the visibility graph of a polygon. Two polygon vertices are connected by an edge if the line segment joining them lies entirely within the polygon. Small perturbations of the polygon can change several visibility relations simultaneously. Prior work has explored triangulation, vertex-based, and signed distance function (SDF) representations for learning polygon realizations from visibility graphs. Triangulation and vertex-based representations explicitly model geometric construction. These representations have shown promising results in mesh generation, where continuous geometric information is available. However, they struggle to preserve the target visibility structure when conditioned only on a discrete visibility graph. A visibility graph contains no distance information between vertices. In contrast, SDF representations provide a continuous distance representation of the polygon and capture the relationship between polygon geometry and visibility more effectively. Existing SDF-based methods start from Gaussian noise and perform diffusion directly in grid space. Their generative process therefore does not explicitly model polygon formation. Our work asks whether the construction-based formulation of geometric representations can be incorporated into SDF-based generative modeling while retaining the strong relationship between SDFs and visibility?

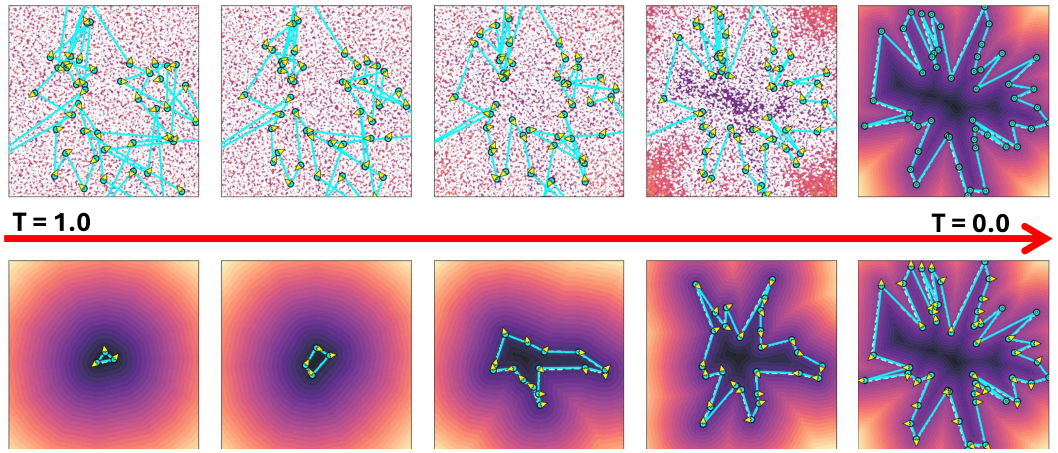


Figure 1: **Rectified Flow vs polygon evolution. Top:** SDF rectified flow starts from Gaussian noise and produces intermediate grid states that are not explicitly tied to stages of polygon formation. **Bottom:** Our formulation starts from a triangle and evolves through intermediate polygon states toward the final polygon.

We present **VisFlow**, a geometrically meaningful flow-matching model that combines a continuous SDF representation with explicit polygon evolution. Standard flow matching typically learns trajectories from an unstructured source distribution, such as Gaussian noise, to the target data distribution. The intermediate states along such trajectories are not tied to stages of polygon formation. Figure 1 illustrates the contrast between such unstructured intermediate states and the explicit polygon evolution used in **VisFlow**. **VisFlow** instead learns an SDF latent representation from polygon trajectories that evolve from an initial triangle to the target polygon. The latent representation is trained so that the intermediate polygon states lie along approximately linear paths. We then use rectified flow to learn the transformation from the initial triangle latent to the target polygon latent, conditioned on the visibility graph. We additionally learn a differentiable Visibility Prediction Network that provides visibility supervision during training and guides optimization of the generated polygon toward the target visibility graph during inference. We further apply the same trajectory-supervision formulation to protein structure generation from contact graphs, demonstrating its use with a different underlying structural evolution process.

Our contributions are as follows:

- We present **VisFlow**, a geometrically meaningful flow-matching model that combines a continuous SDF representation with explicit polygon evolution for visibility-conditioned

polygon generation. We also introduce a Visibility Prediction Network that provides differentiable visibility supervision during training and enables visibility-guided optimization during inference. **VisFlow** improves F1-score by **20%** on *Visibility Reconstruction* and improves coverage by **30%** on *Visibility Characterization* compared with state-of-the-art methods.

- We extend the trajectory-supervision formulation of **VisFlow** to protein structure generation from contact graphs.
- We present a new dataset for *Visibility Reconstruction* with **22%** greater shape diversity than the state-of-the-art dataset.

2 Related Work

We categorize related work into three directions: (i) visibility graph reconstruction, recognition, and characterization, (ii) interpretable flow matching, and (iii) polygon propagation methods.

Visibility Graph Reconstruction, Recognition, and Characterization: Prior work has developed reconstruction and recognition algorithms for restricted polygon classes, including staircase, tower, anchor, spiral, terrain, and convex fan polygons (Abello et al., 1995; Colley et al., 1997; Boomari & Zarei, 2016; Everett & Corneil, 1990; Silva, 2020). However, the general visibility-graph recognition problem remains open (Ghosh & Goswami, 2013). More recently, learning-based methods have modeled the relationship between polygon geometry and visibility. VisDiff (Mahesh et al., 2026) showed that diffusion through an intermediate SDF representation better preserves visibility graphs than GNN- and mesh-based alternatives (Cui & Wei, 2023; Chen et al., 2024; Lionar et al., 2025). However, its coarse grid-based SDF restricts representable polygon geometry, and its intermediate diffusion states do not correspond to a geometry-driven process of polygon formation. We address these limitations using a continuous SDF representation and explicit polygon-evolution supervision.

Flow Matching: Flow matching learns a continuous-time vector field that transports samples from a source distribution to the data distribution (Lipman et al., 2022). Rectified Flow (Liu et al., 2022), a flow-matching formulation, encourages approximately straight source-to-target trajectories. Standard formulations typically use unstructured source distributions such as Gaussian noise, so their intermediate states are not tied to the evolution of the generated structure. Recent work has explored giving flow trajectories physical meaning. Pivi et al. (Pivi et al., 2025) use transitions between equilibrium states of an Ising system, while Singh et al. (Singh et al., 2026) combine flow matching with partially known ODE/PDE dynamics. These approaches incorporate the physical process by changing the dynamics that the flow is trained to follow. In contrast, we use a meaningful geometric evolution to supervise the latent space without replacing the efficient source-to-target transport used for generation by flow matching.

Polygon Propagation Approaches: Polygon propagation defines controlled geometric shape evolution. Grassfire propagation moves the polygon boundary inward along SDF normals (Blum, 1967), while Bruckstein et al. (Bruckstein et al., 1995) update vertices based on their neighbors. Learning-based methods have also used polygon evolution for shape fitting: Deep Snake (Peng et al., 2020) iteratively deforms a polygon toward an object boundary, while Deep Extreme Level Set Evolution (Wang et al., 2019) learns implicit-contour evolution for segmentation. In these methods, the target shape is given and evolution is used to fit its boundary. In contrast, we start from a visibility graph and use polygon evolution to supervise generation of a polygon consistent with the input visibility graph.

3 Problem Formulation

We focus on simple (no boundary crossing), simply connected polygons (i.e., without holes). Let $X \in \mathbb{R}^{N\times 2}$ denote the N vertex coordinates of a polygon P, and let $G \in \mathbb{R}^{N\times N}$ denote its visibility graph represented as an adjacency matrix. The function $V(P)$ computes the visibility graph of P. A visibility graph G is valid if there exists at least one polygon P such that $V(P) = G$.

Visibility Reconstruction: Given a valid visibility graph G, generate a polygon P such that $V(P) = G$.

Visibility Characterization: given a valid visibility graph G, generate all simple polygons P such that $V(P) = G$.

Visibility Recognition: Given a graph G, determine whether there exists a polygon P such that $V(P) = G$.

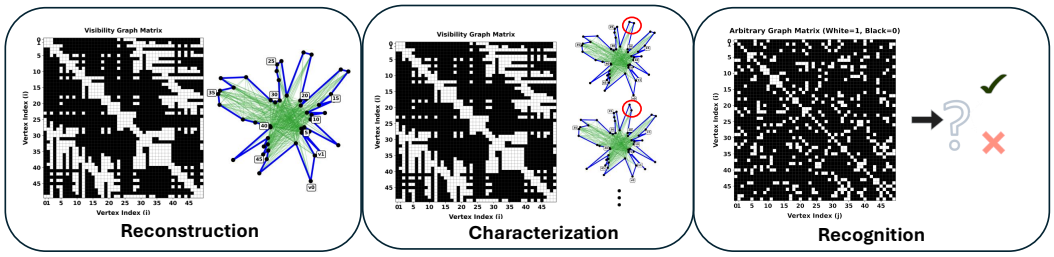


Figure 2: **Visibility graph problems.** Reconstruction finds a polygon consistent with a given visibility graph, characterization finds all polygons corresponding to the same visibility graph (The red circle shows the perturbations but still the same visibility graph), and recognition determines whether an arbitrary graph is a valid visibility graph.

4 METHOD

We now detail the architecture of VisFlow. The central idea is to encode the signed distance function (SDF) of a polygon into spatially localized latent features and learn a latent representation in which stages of polygon formation follow approximately linear trajectories. We construct these trajectories by applying grassfire evolution (Blum, 1967) to progressively collapse each polygon to a triangle and then reversing the resulting trajectory. During sampling, we draw an initial triangle and evolve its latent representation to generate a polygon conditioned on the target visibility graph. VisFlow consists of three main components: an **SDF Latent Generator** that learns spatially localized SDF features and the corresponding polygon-formation trajectories, a **Visibility Prediction Network** that provides differentiable visibility supervision, and a **Flow Network** that models polygon generation conditioned on the target visibility graph. We train the components sequentially: first the Visibility Prediction Network, then the SDF Latent Generator with the visibility network frozen, and finally the Flow Network with both networks frozen. Figure 3 provides an overview of the complete architecture. Grassfire evolution (Blum, 1967) and the visibility formulation motivating our Visibility Prediction Network are detailed in Appendix 9.1.

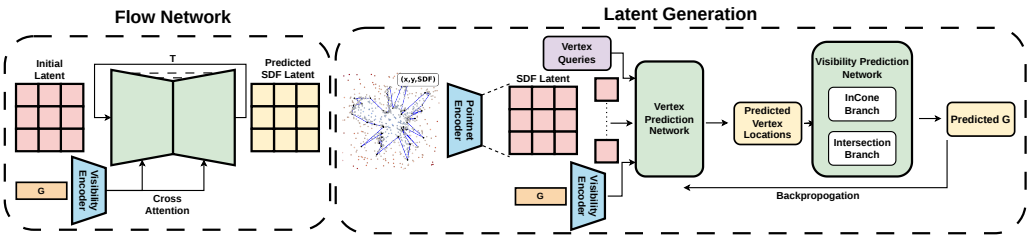


Figure 3: **Overview of VisFlow. Left:** The Flow Network evolves an initial triangle latent into a target polygon latent using a time-conditioned U-Net conditioned on the target visibility graph. **Right:** The SDF Latent Generator encodes a continuous SDF into spatially localized latent features and decodes polygon vertices conditioned on the visibility graph. The Visibility Prediction Network provides differentiable visibility supervision. **Inference:** Given a target visibility graph G, we sample and encode an initial triangle, evolve its latent using the learned flow, decode the final latent into a polygon, and refine its vertices using the Visibility Prediction Network to better match G.

4.1 SDF LATENT GENERATOR

The SDF Latent Generator learns a compact representation of the continuous SDF while preserving the progression of polygon formation. We use a continuous SDF to avoid dependence on a fixed

grid resolution. Direct generative modeling of the continuous SDF would require the flow model to operate over dense spatial SDF samples at every integration step, resulting in high computational and memory costs. We therefore encode the SDF into a compact latent representation. We define polygon-formation trajectories using grassfire flow (Blum, 1967). The inward evolution is stopped when only a triangle remains, the simplest non-degenerate polygonal state. We then reverse the trajectory so that polygon generation proceeds from the triangle to the complete polygon. We normalize trajectory time to $t \in [0, 1]$, with $t = 0$ at the triangle and $t = 1$ at the complete polygon. The resulting grassfire trajectories depend on polygon geometry and therefore follow different deformation paths across shapes. A common latent parameterization is needed to represent these trajectories consistently across polygons. We therefore learn a latent representation in which the trajectories follow approximately straight paths between their endpoint latents. Intermediate grassfire states supervise points along each latent path so that linearly interpolated latents decode to the corresponding stages of polygon formation when conditioned on their visibility graphs.

The straight-line latent trajectories are learned using an encoder–decoder architecture. The encoder represents the input SDF using (x, y, SDF) point samples. We adopt the PatchNets formulation (Tretschk et al., 2020) and represent the encoded SDF using spatially localized patch latents over $[-1, 1]^2$. PointNet (Qi et al., 2017) maps each SDF sample to a feature representation, which is aggregated within its corresponding spatial patch to produce the patch latent. A positional embedding of each patch center encodes its spatial location, and a Transformer encoder exchanges information across patches to produce a context-aware SDF representation. The decoder uses learnable vertex queries to recover polygon vertex locations. The visibility graph G is processed to produce one visibility embedding per vertex, which is concatenated with the corresponding vertex query. A Transformer decoder then cross-attends the resulting vertex representations to the context-aware patch latents and predicts the polygon vertex locations. Additional architecture details are provided in Appendix 9.2.

We train the SDF Latent Generator using reconstruction, alignment, and regularization losses to preserve polygon geometry and visibility while encouraging linear latent trajectories. The overall objective is

$$\mathcal{L}_{\mathrm{latent}} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{ali}}\mathcal{L}_{\mathrm{ali}} + \lambda_{\mathrm{reg}}\mathcal{L}_{\mathrm{reg}}.$$

The alignment loss is

$$\mathcal{L}_{\mathrm{ali}} = \left\| Z_t^{\mathrm{enc}} - \mathrm{sg}\left(\tilde{Z}_t\right) \right\|_2^2, \qquad \tilde{Z}_t = (1-t)Z_{\mathrm{start}} + tZ_{\mathrm{end}}.$$

The reconstruction loss $\mathcal{L}_{\mathrm{rec}} = \mathcal{L}_{\mathrm{coord}} + \mathcal{L}_{\mathrm{vis}}$ combines geometric and visibility supervision. The coordinate loss $\mathcal{L}_{\mathrm{coord}}$ measures the mean squared error between predicted and ground-truth vertices at the start, end, and intermediate states decoded from $\tilde{Z}_t$. Each intermediate state is conditioned on the visibility graph of the corresponding ground-truth grassfire state so that the linearly interpolated latents decode to the corresponding stages of polygon formation. Coordinate reconstruction alone does not enforce the desired visibility relations, so we additionally use $\mathcal{L}_{\mathrm{vis}}$, defined as the binary cross-entropy between the ground-truth visibility graph and the prediction from the frozen Visibility Prediction Network. We next use the alignment loss $\mathcal{L}_{\mathrm{ali}}$ to encourage the encoded grassfire states to follow the linear path between the endpoint latents. The loss encourages the encoded grassfire state Z_t^{enc} to match the corresponding point $\tilde{Z}_t$ on this path, while $\mathrm{sg}(\cdot)$ prevents the endpoint latents from being updated by the alignment objective. Since the alignment loss does not control the magnitude of the latent representation, $\mathcal{L}_{\mathrm{reg}}$ penalizes the mean squared magnitude of the start, end, and encoded intermediate latents. We set $\lambda_{\mathrm{ali}} = 1.0$ and $\lambda_{\mathrm{reg}} = 10^{-3}$ in all experiments.

4.2 Visibility Prediction Network

The Visibility Prediction Network provides a differentiable approximation of polygon visibility. Exact visibility is non-differentiable with respect to polygon vertices, preventing direct visibility supervision during training and gradient-based refinement during inference. We therefore learn a differentiable predictor based on the visibility predicates defined in the preliminaries (Appendix 9.1).

The network first maps each polygon vertex to a positional embedding. For each candidate segment between vertices v_i and v_j, the endpoint embeddings are concatenated to form a candidate-segment representation. The intersection branch compares this representation with each non-incident polygon edge and uses an MLP to predict pairwise intersection probabilities. A max operation aggre-

gates these probabilities to obtain $P_{\text{Intersect}}$, since an intersection with any non-incident edge invalidates visibility. The in-cone branch combines the candidate-segment representation with the neighboring edge representations at each endpoint and uses an MLP to predict the corresponding in-cone probabilities. The two endpoint probabilities are multiplied to obtain P_{InCone}, since the in-cone condition must hold at both endpoints. The final visibility probability is computed as $P_{\text{Vis}} = P_{\text{InCone}}(1 - P_{\text{Intersect}})$. Both branches are trained using binary cross-entropy with their corresponding ground-truth predicate labels.

The pretrained Visibility Prediction Network is frozen during SDF Latent Generator training and used to compute $\mathcal{L}_{\text{vis}}$, allowing visibility gradients to propagate to the predicted polygon vertices. The same network is used during inference for gradient-based refinement toward the target visibility graph. Table 6 reports the effect of the differentiable visibility loss, Table 4 compares the predicate-based architecture with a standard Transformer-based visibility predictor, and Figure 9 shows the complete architecture.

4.3 Flow Network

The Flow Network models the transition from an initial triangle latent to a target polygon latent conditioned on the target visibility graph. We obtain the endpoint latents Z_{start} and Z_{end} by encoding the SDFs of the corresponding trajectory endpoints with the pretrained SDF Latent Generator, which remains frozen during Flow Network training. The approximately linear latent trajectories allow us to model these transitions using rectified flow (Liu et al., 2022). At time t, we construct the linearly interpolated latent as

$$\tilde{Z}_t = (1-t)Z_{\text{start}} + tZ_{\text{end}},$$

with target velocity $Z_{\text{end}} - Z_{\text{start}}$. We sample $t \sim \mathcal{U}(0,1)$ and train the Flow Network with the rectified-flow objective

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_t \left[\left\| v_\theta(\tilde{Z}_t, t, G) - (Z_{\text{end}} - Z_{\text{start}}) \right\|_2^2 \right].$$

The Flow Network predicts the velocity from $\tilde{Z}_t$, time t, and the target visibility graph G. We parameterize the velocity field using a time-conditioned U-Net following Latent Diffusion (Rombach et al., 2022). A separate visibility encoder produces one visibility embedding per vertex, which conditions the U-Net through cross-attention. Additional architecture details are provided in Appendix 9.3.

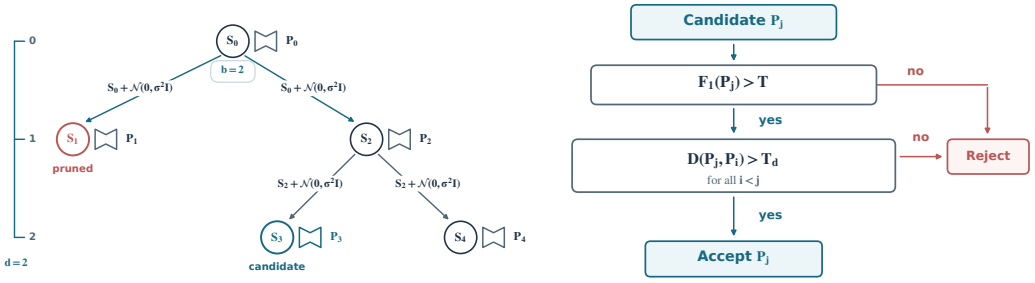
Sampling. Given a target visibility graph G, we sample initial triangles from the valid coordinate range and construct their continuous SDFs. The SDF Latent Generator encodes each triangle SDF to obtain an initial latent. We then integrate the learned velocity field using Euler integration while conditioning on G. The resulting target latent is decoded by the SDF Latent Generator to obtain a polygon. The triangle initialization and approximately linear latent trajectories encourage the intermediate states to follow the corresponding stages of polygon formation. The decoded polygon may not exactly satisfy G, so we refine its vertex locations using binary cross-entropy between the predicted visibility graph and G through the frozen Visibility Prediction Network. We additionally apply binary cross-entropy to predicted intersections between non-incident polygon edges, using zero as the target to penalize self-intersections. The visibility and intersection losses are summed and propagated through the frozen Visibility Prediction Network to optimize the polygon vertices.

5 Results

In this section, we evaluate VisFlow on *Visibility Reconstruction*, *Visibility Characterization*, *Visibility Recognition*, and contact-graph protein structure reconstruction. We first introduce the baselines and evaluation metrics, followed by the results for each problem.

Baselines: We compare VisFlow with Mesh-Generation (Lionar et al., 2025), C-VAE (Harvey et al., 2022), GNN (Cui & Wei, 2023), and VisDiff (Mahesh et al., 2026) on *Visibility Reconstruction*. VisDiff is also used as the baseline for *Visibility Characterization* and *Visibility Recognition*. All methods are trained on the training split of our dataset and evaluated on its test split. Details of the dataset generation and splits are provided in the appendix 9.10

Metrics: We evaluate VisFlow using the metrics defined in VisDiff (Mahesh et al., 2026). The *Visibility Reconstruction* problem is evaluated by comparing the visibility graphs of generated and ground-truth polygons as a binary classification task over vertex pairs. We report accuracy, precision, recall, and F1-score, with F1-score used as the primary metric because of class imbalance. The *Visibility Characterization* problem requires measuring how broadly the space of polygons satisfying a target visibility graph G is covered. Since this space cannot be enumerated exhaustively, we use neighbourhood exploration as a proxy for coverage. We draw a root sample S_0 from the source distribution and generate its corresponding polygon P_0 conditioned on G. We then explore the neighbourhood of S_0 using breadth-first search up to depth d with branching factor b. Each child sample is obtained by applying scheduled perturbations to its parent and is used to generate a polygon conditioned on G. A node is retained only if its visibility F1-score with respect to G exceeds T and its polygon is sufficiently distinct from all previously accepted polygons under the distance threshold T_d. We measure distinctness by aligning the polygons for rotation and translation using Procrustes alignment (Schönemann, 1966), uniformly sampling their boundaries, and computing the Chamfer distance between the sampled points. Coverage is defined as the fraction of explored nodes that are retained. We assess diversity by varying T_d. Figure 4 illustrates the coverage metric.



(a) Exploration Tree Generation

(b) Candidate Acceptance Criteria

Figure 4: **Coverage:** (a) A learned visibility-graph reconstruction network generates polygon P_i from source state S_i. Child states are formed as $S_{\text{child}} = S_{\text{parent}} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. (b) A candidate polygon P_j is accepted if $F_1(P_j) > T$ and $D(P_j, P_i) > T_d$ for every $i < j$. Here, $F_1(P_j)$ is the F1 score between the visibility graph of P_j and the target visibility graph, and $D(P_j, P_i)$ is the Chamfer distance between the two polygons. Coverage is the ratio of expanded nodes to total nodes, $N_{\text{expanded}}/N_{\text{total}}$.

5.1 Visibility Reconstruction

We evaluate VisFlow on *Visibility Reconstruction* against the baseline methods. We generate $N = 50$ polygon samples per generative approach and select the polygon whose predicted visibility graph achieves the highest F1-score with respect to the target graph, while the GNN baseline produces a single polygon. Table 1 shows that VisFlow outperforms all baselines. Table 5 further reports its performance on the original VisDiff dataset, and Figure 10 provides a qualitative comparison. VisDiff exhibits a substantial performance decline on our 50-vertex dataset compared with its original 25-vertex dataset because its coarse SDF resolution limits faithful representation of polygon geometry at higher vertex counts, whereas the continuous SDF representation in VisFlow preserves the geometry. We next examine the source of the performance gains. Table 7 shows that inference-time optimization provides additional improvement, while the flow model already outperforms VisDiff and the other baselines without optimization. We then compare VisFlow with continuous SDF latent diffusion (Cont-VisDiff) in Table 9 to separate the effects of polygon propagation and the continuous SDF latent representation. Polygon propagation provides only a small additional improvement on *Visibility Reconstruction*, indicating that the primary performance gain comes from the continuous SDF representation.

5.2 Visibility Characterization

We evaluate VisFlow on *Visibility Characterization* by measuring coverage under local source-space perturbations. We sample a root noise vector S_0 with variance σ^2 for VisDiff and root triangle parameters T_0 from the valid parameter ranges for VisFlow. Child samples are generated by perturbing their parent samples with Gaussian noise, with σ_p set to 10% of the corresponding sampling scale.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) C-VAE (Harvey et al., 2022)	0.84 ± 0.04	0.78 ± 0.04	0.69 ± 0.03	0.72 ± 0.03
(b) GNN (Cui & Wei, 2023)	0.86 ± 0.06	0.86 ± 0.06	0.69 ± 0.05	0.73 ± 0.06
(c) Mesh (Lionar et al., 2025)	0.84 ± 0.06	0.82 ± 0.11	0.69 ± 0.05	0.71 ± 0.05
(d) VisDiff (Mahesh et al., 2026)	0.86 ± 0.04	0.80 ± 0.04	0.74 ± 0.03	0.76 ± 0.03
(e) VisFlow + Optimization (Ours)	$\mathbf{0.94 \pm 0.03}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$

Table 1: Baseline comparison: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

We use $T_d \in \{0.01, 0.05\}$, $T \in \{0.7, 0.8\}$, $b = 2$, and $d = 5$, with T_d chosen from the minimum and maximum diversity observed among realizations of the same visibility graph in our dataset. Table 2 shows that VisFlow achieves substantially higher coverage than VisDiff, particularly at larger T_d and T. We next analyze the source of this improvement. Table 8 shows that inference-time optimization provides additional gains, while VisFlow already exceeds VisDiff without optimization. We then compare VisFlow with continuous SDF latent diffusion in Table 10 to isolate the effect of polygon propagation. Perturbations of the latent-diffusion source produce few distinct polygons satisfying the target visibility constraint, whereas perturbations of the triangle source in VisFlow produce more polygons satisfying both acceptance criteria. These results indicate that polygon propagation is the primary source of the coverage improvement on *Visibility Characterization*, while the *Visibility Reconstruction* results show that the continuous SDF representation primarily improves visibility reconstruction.

F1 Threshold T	Distance Threshold T_d	VisDiff (Mahesh et al., 2026)	VisFlow + Optimization
0.70	0.01	0.89 ± 0.28	$\mathbf{1.00 \pm 0.00}$
0.70	0.05	0.68 ± 0.29	$\mathbf{1.00 \pm 0.01}$
0.80	0.01	0.03 ± 0.16	$\mathbf{0.95 \pm 0.17}$
0.80	0.05	0.00 ± 0.00	$\mathbf{0.95 \pm 0.19}$

Table 2: Coverage performance (↑) under varying F1 threshold T and distance threshold T_d: each value represents the fraction of nodes expanded by the method relative to the total number of nodes in the BFS search tree.

5.3 Visibility Recognition

We evaluate VisFlow on *Visibility Recognition*. Following VisDiff, we construct the test set by adding holes to half of the polygons in the test split, producing visibility graphs that serve as proxies for invalid inputs. Polygons are generated using the same procedure as in *Visibility Reconstruction*, and an input graph is classified as valid if the visibility F1-score of the selected polygon exceeds threshold TT. Figure 5 compares VisFlow with VisDiff. VisDiff classifies all inputs as invalid, whereas VisFlow achieves 80% recognition accuracy at a threshold near its mean F1-score on *Visibility Reconstruction*. These results show that the improved reconstruction performance on *Visibility Reconstruction* also translates to *Visibility Recognition*.

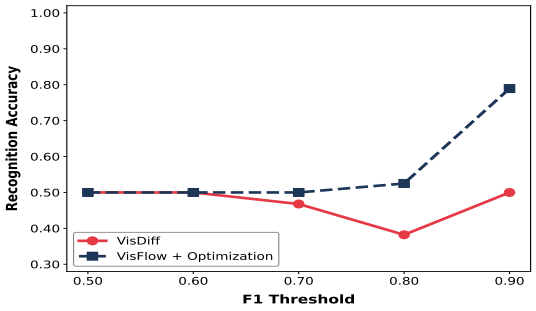


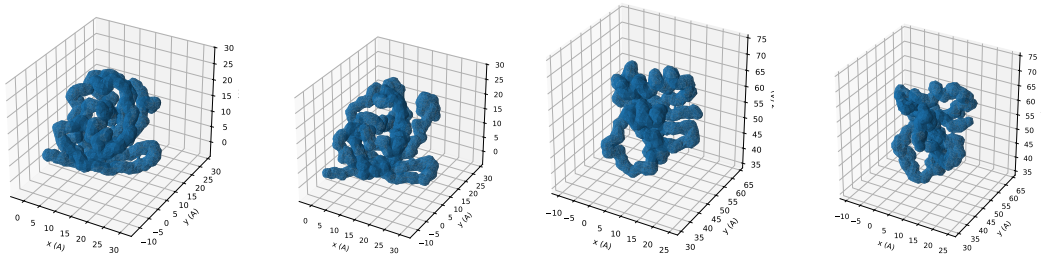
Figure 5: Recognition Performance: Accuracy vs F1 Threshold

5.4 CONTACT-GRAPH RECONSTRUCTION

We next evaluate whether the trajectory-supervision formulation of VisFlow extends beyond polygon reconstruction to protein contact-graph reconstruction. A protein contact graph is a binary adjacency matrix indicating whether residue pairs are spatially close. Since the polygon-formation trajectory does not represent protein conformational dynamics, we use targeted molecular dynamics (Schlitter et al., 1994) trajectories and reverse them to obtain random-coil-to-native reconstruction paths. We replace the polygon-specific encoder and decoder with a contact-graph encoder and a 3D coordinate decoder. The model is trained on ProteinNet (AlQuraishi, 2019) using Kabsch-aligned mean-squared error (Kabsch, 1976) together with a contact-graph reconstruction loss. We evaluate on FragFold150 (Kosciolek & Jones, 2014) by comparing the contact graph induced by the reconstructed structure with the target graph using accuracy, precision, recall, and F1-score. Table 3 shows that VisFlow substantially outperforms the baselines. We omit VisDiff and MeshGPT because their architectures are specialized for 2D polygon and mesh representations and are not directly applicable to 3D protein contact-graph reconstruction. Figure 6 shows qualitative samples generated by VisFlow, and Appendix 9.6 provides architecture details.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) C-VAE (Harvey et al., 2022)	0.79 ± 0.05	0.05 ± 0.02	0.32 ± 0.08	0.08 ± 0.03
(b) GNN (Cui & Wei, 2023)	0.49 ± 0.13	0.06 ± 0.03	0.96 ± 0.03	0.11 ± 0.05
(c) VisFlow(Ours)	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.83 \pm 0.08}$	$\mathbf{0.89 \pm 0.05}$	$\mathbf{0.86 \pm 0.06}$

Table 3: Contact-Graph Reconstruction Performance Fragfold 150 Dataset: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.



(a) GT Protein Sample 1 (b) 0.91 (c) GT Protein Sample 2 (d) 0.92

Figure 6: **Contact-Graph Reconstruction.** Ground-truth protein structures are shown in (a) and (c), with their corresponding VisFlow reconstructions shown in (b) and (d), respectively. The labels report the contact-graph similarity between each ground-truth structure and its reconstruction.

6 CONCLUSION

We presented **VisFlow**, an interpretable flow-matching model that combines continuous SDF representations with explicit polygon evolution for learning geometric realizations from visibility graphs. We show that the continuous SDF representation drives improvements in reconstruction accuracy, while trajectory-supervised polygon evolution leads to substantial gains in coverage and diversity of valid realizations. We introduce a new visibility-graph reconstruction dataset with **22%** greater shape diversity than the state-of-the-art dataset, on which **VisFlow** improves F1-score by **20%** on *Visibility Reconstruction* and coverage by **30%** on *Visibility Characterization*. We further show that the same trajectory-supervision formulation can be applied to protein structure reconstruction from contact graphs. However, our patch-based SDF representation currently assumes that shape information is distributed uniformly across fixed spatial patches. Future work will relax this assumption by introducing the patch radius as an additional parameter, following the original PatchNets (Tretschk et al., 2020) formulation. We also plan to investigate contact-graph realizations as additional structural information for de novo protein generation with AlphaFold (Jumper et al., 2021).

REFERENCES

James Abello, Ömer Egecioglu, and Krishna Kumar. Visibility graphs of staircase polygons and the weak bruhat order, i: from visibility graphs to maximal chains. *Discrete & Computational Geometry*, 14(3):331–358, 1995.

Mohammed AlQuraishi. Proteinnet: a standardized data set for machine learning of protein structure. *BMC bioinformatics*, 20(1):311, 2019.

Thomas Auer and Martin Held. Rpg-heuristics for the generation of random polygons. In *Proc. 8th Canada Conf. Comput. Geom. Ottawa, Canada*, pp. 38–44. Citeseer, 1996.

Harry Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn (ed.), *Models for the Perception of Speech and Visual Form*, pp. 362–380. MIT Press, Cambridge, MA, 1967.

Hossein Boomari and Alireza Zarei. Visibility graphs of anchor polygons. In *Topics in Theoretical Computer Science: The First IFIP WG 1.8 International Conference, TTCS 2015, Tehran, Iran, August 26-28, 2015, Revised Selected Papers 1*, pp. 72–89. Springer, 2016.

Alfred M Bruckstein, Guillermo Sapiro, and Doron Shaked. Evolutions of planar polygons. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(06):991–1014, 1995.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. Meshanything: Artist-created mesh generation with autoregressive transformers, 2024. URL https://arxiv.org/abs/2406.10163.

Paul Colley, Anna Lubiw, and Jeremy Spinrad. Visibility graphs of towers. *Computational Geometry*, 7(3):161–172, 1997.

Guanyu Cui and Zhewei Wei. Mgnn: Graph neural networks inspired by distance geometry problem, 2023. URL https://arxiv.org/abs/2201.12994.

Hazel Everett and Derek G. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11(1):1–26, 1990.

Subir K Ghosh and Partha P Goswami. Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Computing Surveys (CSUR)*, 46(2):1–29, 2013.

William Harvey, Saeid Naderiparizi, and Frank Wood. Conditional image generation by conditioning variational auto-encoders, 2022. URL https://arxiv.org/abs/2102.12037.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5):922–923, 1976.

Tomasz Kosciolek and David T Jones. De novo structure prediction of globular proteins aided by sequence variation-derived contacts. *PloS one*, 9(3):e92197, 2014.

Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26608–26617. IEEE, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Rahul Moorthy Mahesh, Jun-Jee Chao, and Volkan Isler. Visdiff: Sdf-guided polygon generation for visibility reconstruction, characterization and recognition. *Advances in Neural Information Processing Systems*, 38:66549–66580, 2026.

Joseph o'Rourke. *Computational geometry in C*. Cambridge university press, 1998.

Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 8530–8539. IEEE, 2020.

Francesco Pivi, Simone Gazza, Davide Evangelista, Roberto Amadini, and Maurizio Gabbrielli. On the flow matching interpretability. *arXiv preprint arXiv:2510.21210*, 2025.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Jürgen Schlitter, Michael Engels, and Peter Krüger. Targeted molecular dynamics: a new approach for searching pathways of conformational transitions. *Journal of molecular graphics*, 12(2):84–89, 1994.

Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

André C Silva. On Visibility Graphs of Convex Fans and Terrains. *arXiv preprint arXiv:2001.06436*, 2020.

Gurjeet Sangra Singh, Frantzeska Lavda, Giangiacomo Mercatali, and Alexandros Kalousis. Variational grey-box dynamics matching. In Emtiyaz Khan, Yingzhen Li, Arno Solin, and Aaditya Ramdas (eds.), *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 2782–2790. PMLR, 02–05 May 2026. URL `https://proceedings.mlr.press/v300/singh26a.html`.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *European Conference on Computer Vision*, pp. 293–309. Springer, 2020.

Zian Wang, David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Object instance annotation with deep extreme level set evolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7492–7500. IEEE, 2019.

7 AI USE STATEMENT

In this work, we used generative AI tools to support the implementation of evaluated approaches, generation and refinement of plots and visualizations, literature searches for relevant prior work, and grammar and language editing of the manuscript. AI-generated implementation suggestions were reviewed, adapted, and tested by the authors, and papers identified through AI-assisted literature searches were verified using the original sources. All AI-assisted code, figures, and text were checked by the authors before being included in the final manuscript. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

8 Reproducibility Statement

We provide the details required to reproduce our experiments throughout the paper and appendix. The model architecture, training objectives, and sampling procedure are described in the Method section, while the evaluation protocols, baselines, and metrics are specified in the Results section. Dataset generation and split details are provided in Appendix 9.10. Additional details on grassfire evolution and the visibility formulation are given in Appendix 9.1, with the SDF Latent Generator and Flow Network architectures detailed in Appendices 9.2 and 9.3, respectively. The protein contact-graph reconstruction setup is described in Appendix 9.6. The code and dataset are available at `https://drive.google.com/drive/folders/1XHxsZpe2mx-VjA1JoB6HZP_L2z78WD9o?usp=drive_link`.

9 Appendix

We first provide preleminary for our approach in 9.1. Next, we present the architectural details of the SDF Latent Generator in Section 9.2 and the Flow Network in Section 9.3. We then present the architecture and evaluation of the Visibility Prediction Network in Section 9.4. Next, we present additional Visibility Reconstruction results on the original VisDiff dataset in Section 9.5, followed by architectural and training details for contact-graph reconstruction in Section 9.6. We present ablation studies of the training losses and model components in Section 9.7. In Section 9.8, we compare VisFlow with continuous SDF latent diffusion to isolate the contribution of polygon propagation. We then compare the vertex prediction architectures of VisFlow and VisDiff in Section 9.9. Finally, we analyze the statistics and diversity of our generated dataset in Section 9.10.

9.1 Preliminaries

Grassfire Evolution: Grassfire defines a deterministic inward evolution of the polygon boundary in which vertices move along local SDF-normal directions. An edge-collapse event occurs as two neighboring vertices meet and the intervening edge shrinks to zero length, reducing the number of polygon vertices. Figure 7 illustrates the evolution and edge-collapse process.

Visibility Formulation: Two polygon vertices are visible if the segment connecting them lies entirely within the polygon. Visibility decomposes into an in-cone condition at both endpoints and an intersection condition with non-incident polygon edges (o'Rourke, 1998). Visibility between vertices v_i and v_j is defined as

$$V_{ij} = I_{\text{InCone}}(v_i, v_j)\,(1 - I_{\text{Intersect}}(v_i, v_j))\,,$$

where $I_{\text{InCone}} \in 0, 1$ indicates whether $\overline{v_i v_j}$ lies within the valid visibility cone at both endpoints, and $I_{\text{Intersect}} \in 0, 1$ indicates whether it intersects any non-incident polygon edge. Figure 8 illustrates both predicates. The Visibility Prediction Network follows the same decomposition.

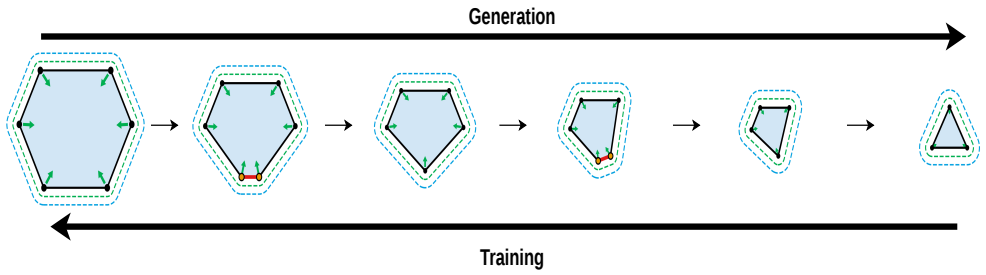
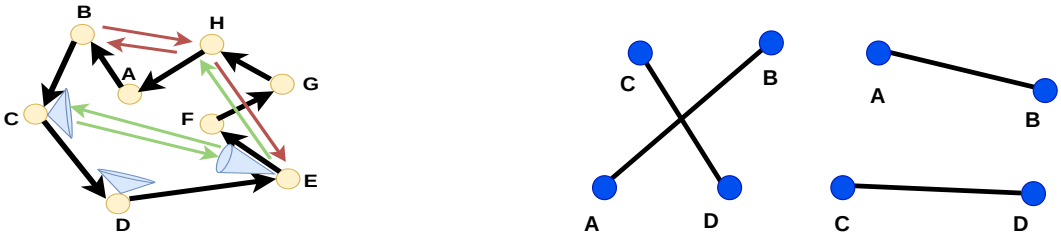


Figure 7: **Grassfire trajectory for polygon formation.** Ground-truth trajectories are constructed by applying grassfire flow to collapse the polygon along local SDF-normal directions until only a triangle remains. The arrows indicate the direction of vertex motion. The yellow vertices mark an edge-collapse event, in which two neighboring vertices meet as the edge between them shrinks to zero length, after which the polygon continues with one fewer vertex. For training, we reverse the trajectory so that polygon formation proceeds from the triangle to the complete polygon. The intermediate grassfire states supervise the corresponding points along the latent trajectory.



(a) In-cone condition (b) Intersection condition

Figure 8: **Visibility predicates.** (a) Two polygon vertices can be visible only if their connecting segment lies within the in-cone region at both endpoints. At each endpoint, the two incident polygon edges define an angular region, and the portion of this region lying inside the polygon forms the local in-cone region, shown in blue. Green arrows indicate that the segment from the other vertex enters through the in-cone region, while red arrows indicate that it enters from outside. Thus, $\overline{BH}$ fails at both endpoints, $\overline{HE}$ satisfies the condition only at H, and $\overline{CE}$ satisfies it at both endpoints. (b) The intersection predicate determines whether a candidate segment intersects a non-incident polygon edge.

9.2 SDF Latent Generator Architecture

In this section, we present the architectural details fo SDF Ltent Generator. We represent each polygon using a spatially structured SDF latent representation. The domain $[-1, 1]^2$ is uniformly partitioned into a 6×6 grid, yielding $N = 36$ spatially localized patches. Each SDF sample is first mapped to a feature representation using PointNet (Qi et al., 2017) and assigned to the patch whose center is nearest to the sample location. Within each patch, the sample features are aggregated using a Gaussian-decayed inverse-distance-weighted average based on their distance to the patch center, producing one latent vector per patch. We use a fixed uniform partition rather than adaptive patch placement to maintain a consistent spatial decomposition across shapes and avoid the additional complexity of adaptive patch optimization (Tretschk et al., 2020). The polygon decoder uses $Q = 50$ learnable vertex queries to recover the polygon vertex locations from the patch latents. The visibility graph G is represented using its upper-triangular entries and processed by an MLP that produces one visibility embedding for each vertex. Each visibility embedding is concatenated with the corresponding vertex query, and the resulting query representations are processed by a Transformer decoder that cross-attends to the patch latents to predict the polygon vertex locations.

9.3 Flow Network Architecture

In this section, we present the architectural details fo Flow Network. The flow network operates on a spatially structured SDF latent representation consisting of 36 patch latents associated with a uniform 6×6 partition of the domain $[-1, 1]^2$. The patch latents are arranged according to their spatial locations to form a 6×6 latent feature map. We parameterize the velocity field using a time-conditioned U-Net following the architecture of Latent Diffusion (Rombach et al., 2022).

The target visibility graph G is represented by its upper-triangular entries. A visibility encoder processes this global graph representation using an MLP and produces one visibility embedding for each vertex. The resulting per-vertex visibility embeddings condition the U-Net through cross-attention, allowing the predicted velocity field to depend on both the current latent state and the target visibility graph.

9.4 Visibility Prediction Network

In this section, we present the detailed architecture of the Visibility Prediction Network and evaluate its performance on ground-truth polygons from the test split of our dataset. Figure 9 illustrates the network architecture. We assess its ability to recover visibility relationships through a comparison with a conventional transformer architecture in Table 4. The results show that our network perfectly reconstructs the visibility graphs of the ground-truth polygons in the test set and substantially outperforms the transformer baseline.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) Visibility Verifier Network	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
(b) Transformer Architecture	0.91 ± 0.02	0.89 ± 0.03	0.76 ± 0.05	0.82 ± 0.04

Table 4: Vertex Prediction comparison: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

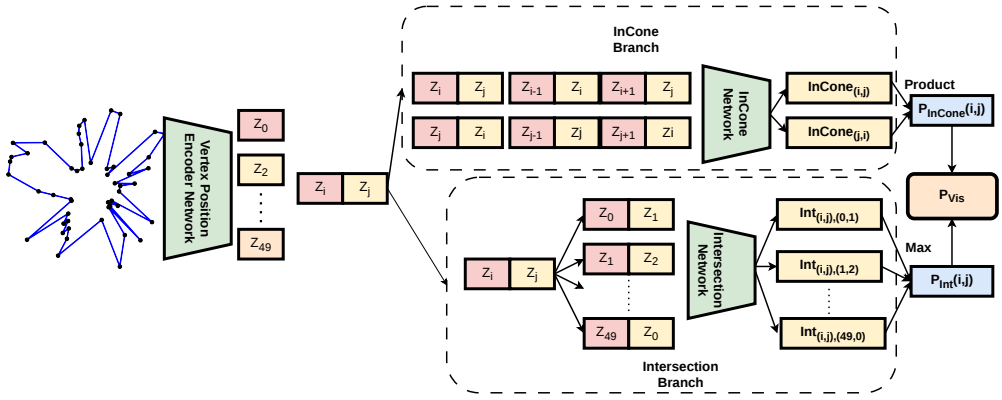


Figure 9: **Architecture of the Visibility Prediction Network.** Polygon vertices are first mapped to positional embeddings by a shared vertex encoder. For a candidate segment (v_i, v_j), the **In-Cone Branch** combines the candidate-segment representation with the neighboring polygon edges at each endpoint and predicts an in-cone probability for both endpoints. The two endpoint probabilities are multiplied to obtain P_{InCone}. The **Intersection Branch** compares the candidate segment with each non-incident polygon boundary edge and predicts pairwise intersection probabilities, which are aggregated using a max operation to obtain $P_{\text{Intersect}}$. The branch outputs are combined as $P_{\text{Vis}} = P_{\text{InCone}}(1 - P_{\text{Intersect}})$ to obtain the final visibility probability for (v_i, v_j).

9.5 *Visibility Reconstruction*

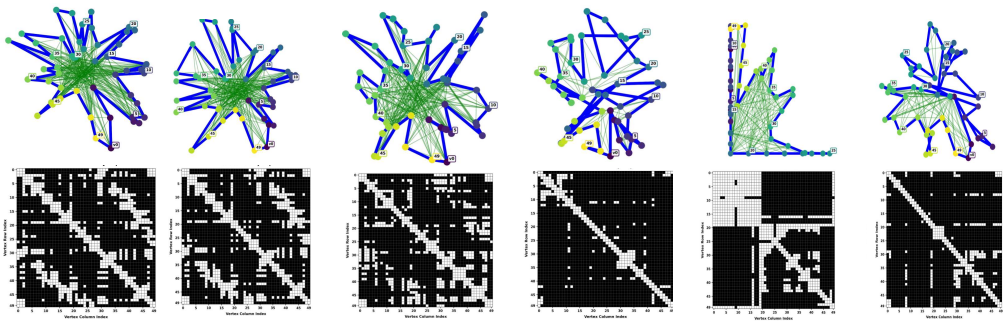
In this section, we additionally evaluate VisFlow on the original VisDiff dataset (Mahesh et al., 2026) and compare its performance with existing baselines. Table 5 shows the comparison between VisFlow and baselines. VisFlow outperforms all baselines showing that the improvements of VisFlow are also retained on the original VisDiff dataset.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) C-VAE (Harvey et al., 2022)	0.74	0.72	0.70	0.70
(b) GNN (Cui & Wei, 2023)	0.73	0.79	0.69	0.67
(c) Mesh (Lionar et al., 2025)	0.75	0.74	0.72	0.71
(d) VisDiff (Mahesh et al., 2026)	0.92	0.91	0.91	0.91
(e) VisFlow + Optimization (Ours)	**0.97**	**0.98**	**0.96**	**0.97**

Table 5: Baseline comparison VisDiff Dataset: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

9.6 Contact-Graph Reconstruction

In this section, we present the architecture details for the contact-graph reconstruction problem. We use the same trajectory-supervised latent-flow formulation as VisFlow, while replacing the polygon-specific components with a protein SDF encoder, a backbone decoder, and a contact-graph encoder. We define the protein contact graph G as a binary adjacency matrix over residues, where $G_{ij} = 1$ if the Cα atoms of residues i and j are within 8 Å, and $G_{ij} = 0$ otherwise.



(a) GT (b) **0.96** (c) 0.75 (d) 0.66 (e) 0.63 (f) 0.70

Figure 10: *Visibility Reconstruction* Qualitative Results: (a) GT, (b)VisFlow + Optimization, (c) VisDiff, (d) GNN, (e) Mesh, (f) C-VAE. The labels below show the F-1 Score comparison with the GT Visibility Graph.

Protein backbones are represented using continuous SDF samples computed from the N, Cα, and C atoms. The SDF encoder uses $N = 64$ patch latent tokens of dimension 128, forming a $128 \times 8 \times 8$ latent representation. The backbone decoder predicts N/Cα/C coordinates for proteins containing up to 270 residues and is conditioned on both the amino-acid sequence and the target contact graph. We use a maximum length of 270 residues because it covers the proteins in the training and evaluation datasets while maintaining a fixed-size representation. The contact graph is represented as a padded 270×270 matrix and encoded row-wise using an MLP followed by a six-layer Transformer. The sequence embeddings and graph features are jointly processed by a six-layer Transformer decoder with hidden dimension 128 and eight attention heads.

The SDF latent representation is trained using the same loss formulation as in polygon generation, with a reconstruction loss $\mathcal{L}_{\text{rec}}$, alignment loss $\mathcal{L}_{\text{ali}}$, and latent regularization loss $\mathcal{L}_{\text{reg}}$. For protein reconstruction, $\mathcal{L}_{\text{rec}}$ consists of a Kabsch-aligned backbone coordinate reconstruction loss $\mathcal{L}_{\text{coord}}$ (Kabsch, 1976) and a contact-graph consistency loss $\mathcal{L}_{\text{contact}}$. Given the predicted C$\alpha$ coordinates $\hat{\mathbf{x}}_i$, we compute the pairwise distances as

$$d_{ij} = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|_2, \tag{1}$$

and define the corresponding contact logits as

$$\ell_{ij} = \frac{8 - d_{ij}}{\tau}, \qquad \tau = 0.5. \tag{2}$$

We define $\mathcal{L}_{\text{contact}}$ as a weighted binary cross-entropy loss between the predicted contacts and the target contact graph for residue pairs satisfying $|i - j| \geq 6$, which excludes short-range sequence contacts and focuses the loss on long-range structural relationships. We use weights 0.05 for contacts and 0.95 for non-contacts. The reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{coord}} + \lambda_{\text{contact}} \mathcal{L}_{\text{contact}}, \tag{3}$$

and the overall latent objective is

$$\mathcal{L}_{\text{latent}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{ali}} \mathcal{L}_{\text{ali}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}. \tag{4}$$

We set $\lambda_{\text{contact}} = 0.01$, $\lambda_{\text{ali}} = 1.0$, and $\lambda_{\text{reg}} = 10^{-3}$.

The SDF encoder and backbone decoder are frozen once the latent representation has been trained. We then train the contact-graph-conditioned Rectified Flow using the same formulation as in polygon generation, with the target contact graph providing the conditioning signal to the latent U-Net. The Rectified Flow model is trained using AdamW with a learning rate of 10^{-4} and weight decay of 10^{-4}, and inference is performed using 60 Euler steps.

9.7 Ablation Study

In this section, we present an ablation study of the training loss on the *Visibility Reconstruction* problem. We also examine the contribution of each module in VisFlow to performance on *Visibility*

Reconstruction and *Visibility Characterization*. Table 6 reports the effect of adding the visibility loss on *Visibility Reconstruction*. The results show that incorporating the visibility loss substantially improves reconstruction performance. Tables 7 and 8 report the effects of adding each module on *Visibility Reconstruction* and *Visibility Characterization*, respectively. The results show that incorporating optimization into VisFlow improves performance on both problems.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) MSE Loss	0.87 ± 0.05	0.81 ± 0.04	0.75 ± 0.03	0.77 ± 0.03
(b) MSE + Visibility Loss	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.84 \pm 0.04}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.83 \pm 0.04}$

Table 6: Loss Ablation Study: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) VisFlow	0.89 ± 0.02	0.84 ± 0.04	0.82 ± 0.04	0.83 ± 0.04
(b) VisFlow + Optimization	$\mathbf{0.94 \pm 0.03}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$
(c) Optimization	0.88 ± 0.04	0.81 ± 0.07	0.88 ± 0.04	0.83 ± 0.06

Table 7: Approach Ablation Study Reconstruction Problem: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

F1 Threshold T	Distance Threshold T_d	VisFlow + Optimization	VisFlow
0.70	0.01	$\mathbf{1.00 \pm 0.00}$	1.00 ± 0.00
0.70	0.05	$\mathbf{1.00 \pm 0.01}$	0.76 ± 0.21
0.80	0.01	$\mathbf{0.95 \pm 0.17}$	0.51 ± 0.47
0.80	0.05	$\mathbf{0.95 \pm 0.19}$	0.36 ± 0.35

Table 8: Approach Ablation Study Characterization Problem: each value represents the fraction of nodes expanded by the method relative to the total number of nodes in the BFS search tree.

9.8 How does VisFlow compare with latent diffusion?

In this section, we examine whether VisFlow offers advantages over a conventional latent diffusion model operating on continuous SDF latents. We compare VisFlow with Cont-VisDiff, which performs diffusion in the SDF latent space without polygon propagation. We evaluate both methods on *Visibility Reconstruction* and *Visibility Characterization*. Table 9 presents the results on *Visibility Reconstruction*. The results show that polygon propagation provides no meaningful improvement in reconstruction performance. Table 10 presents the corresponding results on *Visibility Characterization*. Here, VisFlow achieves substantial gains over Cont-VisDiff. The findings highlight the contribution of polygon propagation to characterization by enabling more effective exploration of distinct polygons consistent with a target visibility graph.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) Cont-VisDiff	0.88 ± 0.02	0.82 ± 0.03	0.83 ± 0.03	0.82 ± 0.03
(b) VisFlow	0.89 ± 0.02	0.84 ± 0.04	0.82 ± 0.04	0.83 ± 0.04
(c) Cont-VisDiff + Optimization	$\mathbf{0.94 \pm 0.02}$	$\mathbf{0.91 \pm 0.03}$	$\mathbf{0.91 \pm 0.03}$	$\mathbf{0.91 \pm 0.03}$
(d) VisFlow + Optimization	$\mathbf{0.94 \pm 0.03}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$

Table 9: Approach Ablation Study: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

F1 Threshold T	Distance Threshold T_d	Cont-VisDiff	VisFlow
0.70	0.01	0.05 ± 0.10	$\mathbf{1.00 \pm 0.00}$
0.70	0.05	0.01 ± 0.00	$\mathbf{0.76 \pm 0.21}$
0.80	0.01	0.05 ± 0.10	$\mathbf{0.51 \pm 0.47}$
0.80	0.05	0.01 ± 0.00	$\mathbf{0.36 \pm 0.35}$

Table 10: Coverage performance (↑) under varying F1 threshold T and distance threshold T_d: each value represents the fraction of nodes expanded by the method relative to the total number of nodes in the BFS search tree.

9.9 Vertex Prediction Architecture

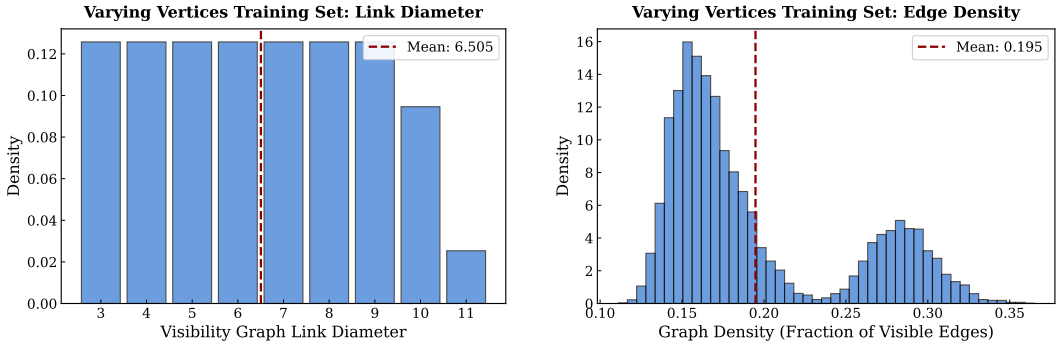
In this section, we compare the vertex prediction architectures of VisFlow and VisDiff. We provide each architecture with the ground-truth SDF and the corresponding visibility graph to assess its ability to recover polygon vertices. Table 11 presents the comparison. The results show that VisFlow recovers polygons more reliably than VisDiff even when both methods receive ground-truth SDFs. The comparison highlights the limitations of VisDiff's coarse SDF resolution for preserving the geometric detail required for accurate vertex recovery, while the continuous SDF representation in VisFlow supports more accurate polygon reconstruction.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) VisDiff (Mahesh et al., 2026)	0.84 ± 0.05	0.78 ± 0.04	0.72 ± 0.03	0.74 ± 0.03
(b) Cont-VisDiff	0.88 ± 0.02	0.81 ± 0.04	0.82 ± 0.03	0.82 ± 0.03
(c) VisFlow	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.83 \pm 0.04}$	$\mathbf{0.83 \pm 0.04}$

Table 11: Vertex Prediction comparison: **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **F1**: F1-Score.

9.10 Dataset Generation

Earlier datasets contain limited variation in polygon concavity. VisDiff broadens the range of concave geometries, but its coarse grid-based SDF limits the geometries that can be represented accurately. We therefore construct a dataset with greater geometric and visibility-graph diversity. We generate a large pool of random polygons using the 2-opt move algorithm (Auer & Held, 1996), normalize them to $[-1, 1]^2$, order their vertices counterclockwise, and canonicalize the ordering by choosing the bottom-left vertex as the first vertex. We compute their visibility graphs and select 10,000 polygons with unique visibility graphs, sampling across link-diameter values to cover a broad range of concavities. For each polygon, we generate 20 additional instances through vertex perturbations that preserve its visibility graph, providing geometrically distinct realizations of the same visibility structure for the one-to-many *Visibility Characterization* problem. The test set uses visibility graphs excluded from training and sampled from the remaining polygon pool, ensuring evaluation on unseen visibility structures. We analyze the generated dataset using link diameter and visibility-graph edge density to characterize its geometric and structural variation. Figure 11 shows the training distributions of these statistics after resampling across link-diameter values. We further compare the range of polygon concavities in our dataset with existing datasets using link diameter. As shown in Table 12, our dataset has a larger average link diameter and covers a wider range of link-diameter values than the compared datasets. In particular, the maximum link diameter increases from 9.0 in VisDiff to 11.0 in our dataset, while the higher average link diameter indicates greater representation of highly concave polygons. These statistics reflect the broader range of polygon structures included in our dataset.



(a) Link Diameter: Train Distribution

(b) Graph Connectivity: Train Distribution

Figure 11: Dataset Statistics: Distribution of (a) polygon link diameter and (b) visibility-graph edge density across polygons in the VisFlow after resampling based on link diameter.

Baselines	Link Diameter (Avg / Std)	Min	Max
MNIST (LeCun, 1998)	1.83 / 0.50	1.0	4.1
COCO 2017 (Lin et al., 2014)	1.32 / 0.25	1.0	3.823
VisDiff (Mahesh et al., 2026)	4.4 / 2.2	1.0	9.0
VisFlow	6.5 / 2.32	1.0	11.0

Table 12: Real-world dataset comparison showing the diversity (in link diameter) between our dataset and standard benchmarks.