VisDiff: SDF-Guided Polygon Generation for Visibility Reconstruction, Characterization and Recognition

Rahul Moorthy
University of Minnesota
mahes092@umn.edu

Jun-Jee Chao
University of Minnesota
chao0107@umn.edu

Volkan Isler
The University of Texas at Austin
isler@cs.utexas.edu

Abstract

The ability to capture rich representations of combinatorial structures has enabled the application of machine learning to tasks such as analysis and generation of floorplans, terrains, images, and animations. Recent work has primarily focused on understanding structures with well-defined features, neighborhoods, or underlying distance metrics, while those lacking such characteristics remain largely unstudied. Examples of these combinatorial structures can be found in polygons, where a small change in the vertex locations causes a significant rearrangement of the combinatorial structure, expressed as a visibility or triangulation graphs. Current representation learning approaches fail to capture structures without well-defined features and distance metrics.

In this paper, we study the open problem of *Visibility Reconstruction*: Given a visibility graph G, construct a polygon P whose visibility graph is G. We introduce **VisDiff**, a novel diffusion-based approach to generate polygon P from the input visibility graph G. The main novelty of our approach is that, rather than generating the polygon's vertex set directly, we first estimate the signed distance function (SDF) associated with the polygon. The SDF is then used to extract the vertex location representing the final polygon. We show that going through the SDF allows **VisDiff** to learn the visibility relationship much more effectively than generating vertex locations directly. In order to train **VisDiff**, we create a carefully curated dataset. We use this dataset to benchmark our method and achieve **26%** improvement in F1- Score over standard methods as well as state of the art approaches. We also provide preliminary results on the harder visibility graph recognition problem in which the input G is not guaranteed to be a visibility graph. To demonstrate the applicability of VisDiff beyond visibility graphs, we extend it to the related combinatorial structure of triangulation graph. Lastly, leveraging these capabilities, we show that VisDiff can perform high-diversity sampling over the space of all polygons. In particular, we highlight its ability to perform both polygon-to-polygon interpolation and graph-to-graph interpolation, enabling diverse sampling across the polygon space.

1 Introduction

Polygons are widely used as geometric representations in domains such as cartography [1], architectural design [2], and robotic motion planning [3]. Applications across these domains require understanding the combinatorial structure of polygons for analysis, reasoning, and generation. The combinatorial structure captures the discrete relationships between polygon vertices, independent of their specific coordinates, angles, or edge lengths. A key example of such a structure is the visibility graph (Appendix A), which encodes mutual visibility between regions. Visibility graphs enable privacy-aware floorplan design [2] and the extraction of topographic features in terrain analysis [4]. While generative models are increasingly used in these applications [5]. Most existing

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

approaches [6, 7] rely solely on geometric coordinates. Despite the structural importance of visibility graphs, they have not been leveraged to guide the generative process. To address this gap, we investigate representations that link polygons to their combinatorial structures.

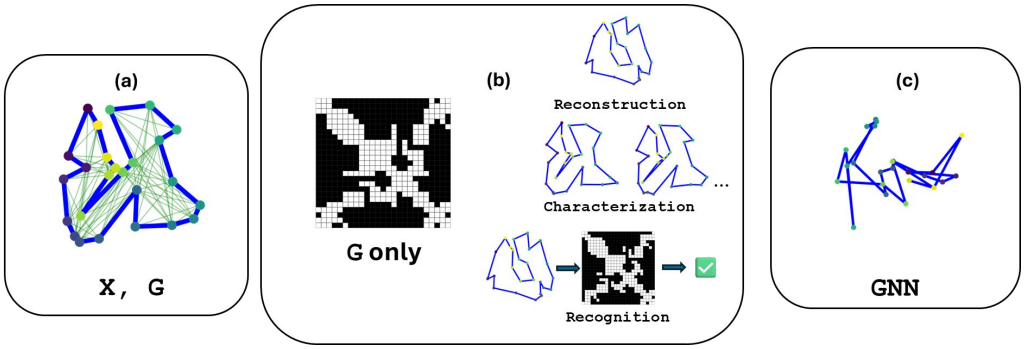


Figure 1: **a)**: A polygon P is given by an ordered list of vertex locations X. Also shown are the visible edges of the polygon in green. **b)**: The visibility graph G of polygon P represented as an adjacency matrix where black denote non-visible edge while white denote visible edges. We seek to answer the question: How much information about X can be recovered from G alone? We show the output of VisDiff for the reconstruction, characterization and recognition problems associated with G. **c)**: GNN output of G for reconstruction problem. Clearly, standard GNN embedding methods are not sufficient to recover the vertex locations X from G.

The main questions we study are as follows: Suppose we are given the visibility graph G of a polygon, as shown in Figure 1b. Note that G contains no coordinate information from X. What can we say about the polygon (Reconstruction) and the set of all polygons (Characterization) that have this graph G as their visibility graph. Can we determine whether a polygon exists for G (Recognition). Given the recent success of learning-based graph-embedding approaches, it might be tempting to apply GNN-based methods [8] directly to reconstruct X. However, since G lacks a natural distance metric that such models can exploit, these methods fail to directly predict vertex coordinates from G (Figure 1c).

In this paper, to overcome the lack of distance information in G for polygon generation, we present **VisDiff**: a generative diffusion model that employs an intermediate signed distance function (SDF) representation. Specifically, given a visibility graph G as input, VisDiff generates the SDF of the corresponding polygon. The vertex locations are then extracted from the zero level set of the SDF, yielding the polygon associated with G. Our core contribution is that using the SDF as an intermediate representation enables the generation of meaningful polygons that preserve visibility constraints and provide strong evidence for all characterization, reconstruction, and recognition as shown in Figure 1b. This contrasts with prior approaches that rely directly on vertex [8] or triangulation [9] representations. To train VisDiff, we construct a carefully curated dataset that captures a broad range of polygon combinatorial properties. Existing random polygon generation methods struggle to faithfully represent the visibility graph space, often biasing toward high concavity as the number of vertices increases. We address this issue by systematically rebalancing the dataset by link diameter, which quantifies concavity. The dataset is made publicly available for further research.

We demonstrate the generality of VisDiff beyond visibility graphs. Specifically, we apply it to the task of reconstructing a polygon from its triangulation graph. Finally, we leverage these capabilities to highlight its ability to perform high-diversity sampling over the space of all polygons, making it suitable for data augmentation in applications involving polygon representations.

In summary our key contributions are:

- We initiate a learning based study of polygon reconstruction and characterization problems based on combinatorial structures without well-defined distance and neighborhood. We show that existing state-of-the-art approaches fail to effectively capture the connection between combinatorial and geometric properties.
- We design a carefully curated dataset that captures a wide range of combinatorial properties of polygons and make it publicly available for further research.

- We present **VisDiff**: a generative diffusion model that generates an intermediate SDF representation corresponding to G, which is then used to extract the polygon. We show that using SDF as an intermediate representation yields meaningful polygons for G. We evaluate VisDiff with baselines on *Visibility Reconstruction* and demonstrate its capability to perform *Visibility Characterization*. Additionally, we also provide preliminary results for *Visibility Recognition*.
- We demonstrate the generality of **VisDiff** to both visibility graph and triangulation graph based polygon generation.
- We leverage all these capabilities to highlight the ability to perform high diversity sampling over the space of all polygons making it suitable for applications such as data augmentation.

2 Related Work

We summarize the related work in three directions: theoretical results for visibility graph reconstruction and recognition, representation learning for shapes, and graph neural networks for polygons.

Visibility Graph Reconstruction and Recognition: The problems of reconstructing and recognizing visibility graphs have been studied extensively in the theoretical computational geometry literature, yet they remain open [10]. Existing results address reconstruction and recognition only for specific polygon categories, including pseudo [11], convex fan [12], terrain [12], spiral [13], anchor [14], and tower [15] polygons. On the hardness side, the visibility graph recognition and reconstruction problems are known to lie in PSPACE [16], specifically within the Existential Theory of the Reals class [17]. However, the exact computational hardness of these problems remains unresolved. In this work, we explore them from a representation learning perspective, investigating whether generative models can learn the underlying manifold of the space of polygons and their visibility graphs in a generalizable manner.

Representation Learning: 3D shape completion [18, 19, 20, 21] is a closely related application. In 3D shape completion, the input typically contains partial geometric information, such as a point cloud. In contrast, our input consists solely of a combinatorial description, namely the visibility graph. Multiple shapes may be consistent with the same input graph, and recovering them without any geometric cues is a fundamentally challenging task. Another related body of work is mesh generation [6, 7]. Recent advances in this area include MeshGPT [22], MeshAnything [9], and PolyDiff [23]. These approaches generate high-quality 3D triangular meshes by learning to output a set of triangles from a fixed set of triangles. PolyDiff discretizes the 3D space into bins, whereas MeshAnything and MeshGPT operate over a predefined set of triangles. In contrast, our work seeks to learn the continuous space of all polygons and their corresponding visibility graphs.

Graph Neural Networks (GNNs): GNNs are a standard class of models used for learning on graph-structured data. Most existing work focuses on graphs with features embedded in a well-defined metric space. The closest to our setting is the generation of graph embeddings from distance matrices. Cui et al. [24] proposed MetricGNN, which learns graph embeddings from a given embedding distance matrix. Yu et al. [25] introduced PolygonGNN, which effectively represents multi-polygon data for graph classification tasks by leveraging visibility relationships between polygons. Specifically, PolygonGNN demonstrated that augmenting vertex embeddings of individual polygons with both spatial location information and visibility relationships to other vertices leads to improved geometric representation learning. All of the above approaches assume the existence of an underlying metric space or spatial position information, both of which are absent in the visibility graph reconstruction problem. We develop **VisDiff** to learn meaningful embeddings in this challenging combinatorial domain.

3 Problem Formulation

We only study polygons which are simple (the boundary does not self intersect) and simply-connected (no holes). Let $X \in \mathbb{R}^{N\times 2}$ be the N vertex locations of a polygon P, $G \in \mathbb{R}^{N\times N}$ be the adjacency matrix representing visibility graph of P, $Vis(P)$ be a function to determine the visibility graph of P as an adjacency matrix. We consider the following problems of increasing difficulty:

Problem 1 (Reconstruction) *Given a valid G, generate* ***a*** *polygon P such that $Vis(P) = G$.*

Problem 2 (Characterization) *Given a valid G, generate **all** polygons P such that $Vis(P) = G$.*

Note that in these two problems, the input G is assumed to be valid – i.e., there exists a polygon P whose visibility graph is G. We also formulate a more general recognition problem in which G is arbitrary:

Problem 3 (Recognition) *Given an arbitrary graph G, determine whether there exists a polygon P such that $Vis(P) = G$.*

4 Method

We present VisDiff for generating a polygon distribution given the visibility graph. VisDiff models the polygon distribution using diffusion models [26] conditioned on the input graph. The key idea is to use the signed distance function (SDF) as an intermediate representation for polygon generation. In this section, we first provide a brief background on diffusion models, then introduce the two main components of VisDiff: (i) a graph-conditioned diffusion model for SDF generation, and (ii) a polygon vertex extraction module that reconstructs the polygon from the predicted SDF. Detailed architecture specifications are provided in Appendix J.

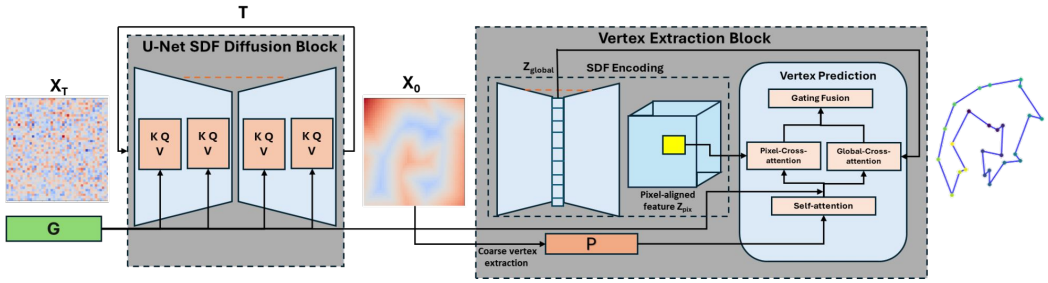


Figure 2: VisDiff architecture. The model consists of two main components: the U-Net SDF Diffusion block and the Vertex Extraction block. **U-Net Diffusion Block:** A noisy SDF, denoted as $\mathbf{X}_T$, is first sampled from a Gaussian distribution. $\mathbf{X}_T$ then passes through $\mathbf{T}$ timesteps of the reverse diffusion process to generate the clean SDF $\mathbf{X}_0$. This denoising process is conditioned on the input graph $\mathbf{G}$ using transformer cross-attention blocks represented by $\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$, which correspond to the key, query, and value terms, respectively. In our approach, $\mathbf{Q}$ is obtained from the learned spatial CNN features, while $\mathbf{K}$ and $\mathbf{V}$ are derived from $\mathbf{G}$. An initial set of vertices $\mathbf{P}$ is then estimated from $\mathbf{X}_0$ via contour extraction. **Vertex Extraction Block:** Given the predicted SDF $\mathbf{X}_0$, the SDF encoder generates pixel-aligned features Z_{pix} and global features Z_{global}. These features, along with the initial vertices $\mathbf{P}$, are fed into the vertex prediction block to predict the final vertex locations. During **Training**, the model is supervised using both the ground-truth SDF and the corresponding polygon. During **Testing**, only the visibility graph $\mathbf{G}$ is provided as input.

4.1 Background

Diffusion models have demonstrated the ability to efficiently map a Gaussian distribution to a target data distribution [26, 27, 28, 29]. The Denoising Diffusion Implicit Model (DDIM) [30] consists of two main stages: the forward diffusion process and the reverse denoising process.

Forward diffusion involves progressively adding noise to the data according to a predefined schedule. Let the data sample from the target distribution be denoted as x_0. At diffusion timestep t, given a noise standard deviation $\sigma_t > 0$, the noisy sample is defined as

$$x_t = x_0 + \sigma_t \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$ is Gaussian noise. In this way, noise is gradually injected into the data until it becomes a pure Gaussian sample at the end of the forward process. VisDiff employs a linear-log scheduler [31] to control the noise level throughout this process.

Reverse diffusion involves recovering the original data from the final Gaussian sample produced during the forward process. In this step, we start with Gaussian noise and iteratively predict the noise

added to the sample given σ_t. The reverse process is parameterized by a neural network that learns to predict the added noise from the noisy sample and the corresponding σ_t.

4.2 Graph conditioned SDF diffusion

The core idea of VisDiff is to predict the polygon distribution conditioned on the input graph. However, such a graph contains no coordinate information, posing a challenge for existing methods that attempt to directly predict polygon vertex coordinates. To obtain the ground-truth SDF, we first normalize each polygon to fit within a unit square and compute the SDF values on a 40×40 grid, resulting in an image where each pixel stores the distance to the nearest point on the polygon boundary. In this manner, polygons are represented by their signed distance functions as images.

We adopt an architecture similar to Latent Diffusion [32] due to its ability to produce high-quality generations under external conditioning. However, unlike Latent Diffusion, we train directly on the SDF rather than on latent features. Specifically, we use a time-conditioned U-Net [33] encoder–decoder architecture to predict the noise added to the original SDF sample. The U-Net CNN blocks are conditioned on the encoded visibility graph using Spatial Transformer Cross-Attention [34] blocks, which integrate visibility information into the U-Net's spatial features during training. The key and value components of each cross-attention block correspond to the lower-triangular part of the visibility adjacency matrix, since it is symmetric in nature, while the queries are derived from the spatial CNN features. Figure 2 illustrates the architecture of the SDF diffusion block. The model is trained using L_{MSE} mean-squared error loss between the predicted and true noise added to the sample. Given a visibility graph G, the trained model is then used to sample polygon SDFs.

Sampling of the SDF is performed using a DDIM sampler. The process begins by drawing a sample $x_t \in \mathbb{R}^{40\times40}$ from a Gaussian distribution $\mathcal{N}(0, I)$, followed by a schedule of decreasing noise levels proportional to the number of diffusion steps. Each reverse step is defined as

$$x_{t-1} = x_t + (\sigma_{t-1} - \sigma_t)\, \epsilon_\theta(x_t, \sigma_t, G), \tag{2}$$

where $\epsilon_\theta(x_t, \sigma_t, G)$ denotes the noise predicted by the U-Net encoder–decoder architecture, conditioned on the visibility graph G, the current noisy sample x_t, and the corresponding noise level σ_t. This iterative process reconstructs the polygon's SDF while preserving the visibility constraints imposed by G.

4.3 Vertex Extraction

The generated SDF of the polygon is then used to determine the final vertex locations whose visibility relationships correspond to the visibility graph G. The process of selecting vertex locations along the zero level set is challenging, as polygon corners are often ill-defined in the SDF representation. Furthermore, as the number of vertex locations increases, a small change in the placement of points on the SDF will significantly alter the visibility structure of the entire polygon.

We formulate polygon vertex extraction as a separate estimation problem: determining vertex locations given the SDF and the visibility graph G. The vertex extraction architecture comprises two modules: **SDF Encoding** and the **Vertex Prediction Block**. Figure 2 illustrates the architecture of the vertex prediction block.

SDF Encoding: Understanding the fine-resolution structure of the SDF is essential for extracting the underlying polygon representation given G, as small perturbations in point placement on the SDF can significantly alter the visibility structure of the entire polygon. Previous continuous-coordinate polygon extraction methods [35, 36, 37] from images typically rely only on global feature extractors. In contrast, pixel-aligned features have proven highly effective for capturing fine-grained details in tasks such as object detection [38] and 3D reconstruction [39]. Since our polygon extraction from SDF requires fine-grained spatial information, we adopt a PIFu-inspired [39] architecture to extract both pixel-level and global features for encoding the SDF. Specifically, we train a U-Net to encode the SDF into a pixel-aligned embedding space $Z_{\text{pix}} \in \mathbb{R}^{40\times40\times128}$ and a global embedding $Z_{\text{global}} \in \mathbb{R}^{25\times512}$. The generated SDF features are then passed to the vertex prediction block to estimate the ordered vertex locations of the polygon.

Vertex Prediction: We adopt an architecture similar to Polyformer [40] to generate the final polygon from the encoded SDF. Previous studies have shown that polygon initialization improves localization

accuracy compared to random initialization or the use of learnable queries [37]. Therefore, the vertex locations of the polygon P are first extracted using contour detection methods and simplified to 25 vertices using the area-based simplification technique of Visvalingam [41]. The simplified vertex locations are converted into positional embeddings of size 256, and ordering-based positional embeddings are added to capture cyclic ordering. These vertex embeddings, together with Z_{global}, Z_{pix}, and G, are then used to predict the final vertex locations.

Specifically, the vertex embeddings are used as queries Q in three layers of transformer encoders to refine features based on Z_{global}, Z_{pix}, and G. Each encoder layer performs self-attention, where the vertex embeddings serve as keys K and values V. The self-attention is followed by cross-attention, where K and V are formed by concatenating Z_{global} and Z_{pix} with G. The pixel-aligned feature map Z_{pix} is extracted only at the vertex locations P using bilinear interpolation. To adaptively balance the contributions of Z_{global} and Z_{pix}, we employ a sigmoid-gating fusion mechanism [42] to combine their cross-attention outputs. Figure 2 illustrates the structure of each transformer encoder block. An ablation study (Appendix C) shows that combining pixel-level and global features yields significantly more accurate polygon reconstructions than using global features alone for the visibility reconstruction task.

Training: The vertex prediction and SDF encoding blocks are jointly trained using an L_{MSE} loss, which penalizes deviations of the predicted vertex locations from the ground-truth positions. The vertex extraction and SDF diffusion blocks are trained in two stages. In the first stage, only the SDF diffusion block is trained. In the second stage, the SDF diffusion block is frozen, and only the vertex extraction block is trained. This two-stage training strategy is necessary because contour initialization and simplification are non-differentiable operations, which prevent gradient propagation through the entire architecture during joint training.

5 Dataset Generation

The *Visibility Characterization* and *Visibility Reconstruction* problems require a dataset distribution with a key property: multiple polygons P corresponding to the same visibility graph G. In addition, the dataset should exhibit high diversity across different visibility graphs. Since no existing dataset satisfies these criteria, we construct one by uniformly sampling polygons based on the graph properties described below and generating multiple augmentations of each polygon.

The dataset generation process involves sampling 60,000 polygons, each with 25 vertex locations arranged in a fixed anticlockwise order. The vertex coordinates are drawn from a uniform distribution within $[-1, 1]^2$. We employ the 2-opt move algorithm [43] to generate valid polygons from the sampled locations. However, the resulting dataset exhibits non-uniformity with respect to the link diameter of the visibility graph, where link diameter quantifies the maximum number of edges on the shortest path between any two graph nodes. A higher diameter indicates greater polygon concavity. To achieve a balanced distribution, we resample the dataset based on the link diameter of the visibility graph, resulting in a final subset of 18,500 polygons. Additional statistics in Appendix B.1 (Figure 5b) show that our dataset is uniformly distributed in terms of link diameter.

We further augment each polygon to generate 20 samples per visibility graph G. Shear transformations and vertex perturbations are applied while preserving the visibility graph structure. These augmentations introduce the property of multiple polygons sharing the same combinatorial graph G. Both augmentation and resampling are essential for learning the representative space of the *Visibility Characterization* and *Visibility Reconstruction* problems. The final training dataset consists of 370,000 polygons along with their corresponding visibility and triangulation graphs.

The test dataset for validating our approach is generated in two splits: *in-distribution* and *out-of-distribution*. In-distribution samples are obtained by setting aside 100 unique polygons per link diameter from the larger dataset, ensuring they are not included in the training set. Out-of-distribution samples are generated using specific polygon types whose visibility graph properties differ significantly from those in the training set. Appendix B.2 provides details of the out-of-distribution test set generation process and further demonstrates that our dataset exhibits greater diversity in visibility graph properties compared to existing real-world datasets. Both the training and testing datasets are publicly available for further research.

6 Experiments

We compare VisDiff with existing methods on the *Visibility Reconstruction* problem and further demonstrate its effectiveness on the *Visibility Characterization* problem. We also present preliminary results for the *Visibility Recognition* problem. In addition, we evaluate the generalization capability of VisDiff to other combinatorial graph structures, such as triangulation graphs. Finally, we demonstrate the ability of VisDiff to perform high-diversity data sampling over the space of all polygons.

6.1 Experimental Setup

Metrics: To evaluate our algorithm, we compute the visibility graphs of the generated polygons and compare them with the ground truth. We formulate this as a binary classification problem, where each edge in the visibility graph is classified as either *visible* or *non-visible*. We report accuracy, precision, recall, and F1-score between the generated and ground-truth visibility graphs. Each visibility graph is evaluated individually, and the average performance over the dataset is reported as the collective quantitative metric. Since the ratio of visible to non-visible edges can vary significantly across polygons, we primarily use the F1-score to assess model performance.

Baselines: We compare VisDiff against baselines capable of conditional polygon generation using either vertex-based or triangulation-based representations. Specifically, we evaluate state-of-the-art approaches including MeshAnything (Mesh) [9], Vertex-Diffusion (VD) [44], Conditional-VAE (C-VAE) [45], and GNN-based generation [46]. We use the publicly available implementations of MeshAnything and Conditional-VAE. MeshAnything is modified to operate on 2D polygon triangulation representations instead of 3D meshes. For the GNN and Vertex-Diffusion baselines, we adopt the architectures from MGNN [46] and PolyDiff [44], respectively. All baselines are trained on our dataset to ensure fairness in evaluation.

6.2 Visibility Reconstruction

We demonstrate the ability of VisDiff to learn meaningful polygon representations from visibility graphs on the *Visibility Reconstruction* problem. Table 1 reports the quantitative evaluation on the in-distribution dataset. MeshAnything [9] often generates disconnected triangle sets. Therefore, we report results only for polygons forming a closed polygonal chain. It can be observed that VisDiff outperforms Conditional-VAE and Vertex-Diffusion, which rely on vertex representations, and MeshAnything, which uses a triangulation-based representation, by leveraging the intermediate SDF representation across all metrics. Appendix D (Figure 8) presents additional qualitative comparisons of the baselines with VisDiff. VisDiff learns to generate polygons that closely match the ground-truth visibility, whereas the baselines often produce invalid polygons. Out-of-distribution results, visibility reconstruction selection strategy, and further qualitative examples are provided in Appendix D.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑	EDist ↓
(a) VD [44]	0.777	0.773	0.716	0.724	0.44
(b) C-VAE [45]	0.74	0.718	0.704	0.702	0.381
(c) GNN [46]	0.73	0.786	0.686	0.674	0.531
(d) Ours	**0.924**	**0.914**	**0.911**	**0.912**	0.277
(e) Mesh [9]	0.747	0.739	0.723	0.712	**0.269**

Table 1: Baseline comparison: (a) Vertex-Diffusion, (b) Conditional-VAE, (c) GNN, (d) VisDiff, (e) MeshAnything. **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall, **EDist**: Euclidean distance between point sets for triangulation evaluation.

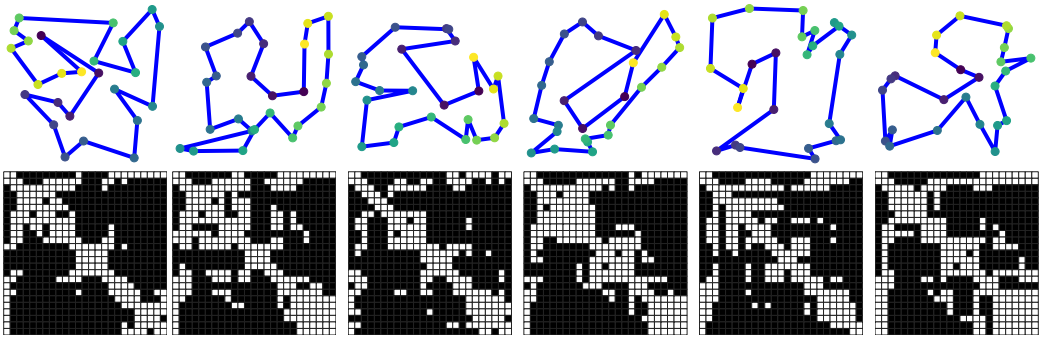
6.3 Visibility Characterization

VisDiff can address the *Visibility Characterization* problem by varying the diffusion process seeds. Figure 3 illustrates how VisDiff produces distinct polygons with varying perturbations while preserving visibility consistent with the ground-truth graph G. Additional qualitative results are provided in Appendix E.

We introduce two metrics, coverage and diversity, to quantitatively evaluate the *Visibility Characterization* problem.

The diversity for a given visibility graph G is computed by sampling N valid polygons using the recognition method in Section F and calculating pairwise Chamfer distances between their point sets. The Chamfer distance captures geometric variation while remaining invariant to vertex ordering. VisDiff achieves a mean Chamfer distance of 0.56 across the test set with $N = 50$, which corresponds to roughly 20% of the 2×2 domain and indicates substantial diversity among the generated polygons.

The coverage metric measures the extent of latent space exploration for a given visibility graph G. We propose a metric-based exploration algorithm to evaluate this coverage. We initialize the root polygon P_0 by sampling a latent noise vector with base standard deviation σ and generating a polygon using VisDiff. A breadth-first exploration is then performed up to a fixed depth d and branching factor b, where each child is generated by adding scheduled noise to its parent in latent space. A node is expanded only if (i) the generated polygon is valid with an F1-score above threshold T using the recognition method in Section F, and (ii) it is distinct from previously discovered nodes based on distance threshold T_d. The coverage metric is defined as the ratio of expanded nodes to the maximum possible nodes, representing the fraction of the latent neighborhood explored. Table 2 reports the performance of VisDiff across different hyperparameters. On average, VisDiff expands about 51% of possible nodes across the test set, given 20 training augmentations per visibility graph. This demonstrates that the latent exploration strategy effectively discovers a broader set of valid solutions than those encountered during training.



(a) GT (b) 0.827 (c) 0.811 (d) 0.827 (e) 0.8 (f) 0.811

Figure 3: *Visibility Characterization*: The top row shows multiple polygons generated by VisDiff for the same visibility graph G. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows the visibility graph corresponding to the polygons where **white** denote visible edge and **black** denote non-visible edge. The caption shows the F1-Score compared to the ground truth (GT) visibility graph.

F1 Threshold T	Depth d	Branching Factor b	Distance Threshold T_d	Coverage Metric ↑
0.85	5	2	0.1	0.475
0.80	5	2	0.1	0.488
0.75	5	2	0.1	0.495
0.70	5	2	0.1	**0.515**

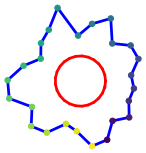
Table 2: Coverage metric for different hyperparameters. Higher coverage indicates broader exploration of the latent space.

6.4 Visibility Recognition

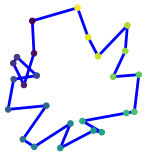
We present preliminary results on the *Visibility Recognition* problem. We generate a test set of 50 valid and non-valid visibility graphs for this task. Polygons with holes are used as examples of non-valid visibility graphs. A polygon with a hole has both an outer boundary and one or more inner boundaries, making it a non-simple polygon. The visibility graph is computed in the same way as for

simple polygons. However, any edge passing through a hole is considered non-visible, as the hole region lies outside the polygon.

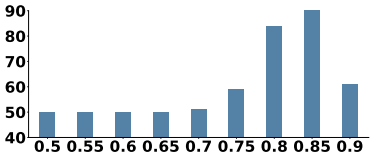
To determine whether a given visibility graph G is valid, we first sample a set of polygons S from G using VisDiff. G is classified as a valid graph if any polygon in S is valid and achieves an F1-score above a predefined threshold X. Figure 4b presents qualitative results of polygon generation by VisDiff for a non-valid visibility graph. In this case, VisDiff fails to generate any valid polygon with an F1-score above 0.85, leading to its classification as a non-valid visibility graph. Figure 4c shows the performance of our model on the *Visibility Recognition* problem under different F1-score thresholds. VisDiff correctly classifies 90% of the samples from the set of valid and non-valid visibility graphs when the F1 threshold is set near the mean performance observed on the *Visibility Reconstruction* problem. This classification accuracy demonstrates that VisDiff effectively captures and represents the underlying space of valid visibility graphs. Additional qualitative results for the *Visibility Recognition* problem are provided in Appendix F.



(a) Non-Valid Sample: Red represents hole



(b) VisDiff prediction given non-valid graph



(c) Accuracy (%) vs F-1 Threshold

Figure 4: We provide qualitative and quantitative results of VisDiff on *Visibility Recognition* problem.

6.5 Triangulation

In this section, we replace the visibility graph input with the triangulation graph to demonstrate the versatility of VisDiff for the reconstruction problem. Note that a polygon may admit multiple valid triangulations, each consisting of $n-2$ triangles, where n is the number of vertices [47]. We employ the Constrained Delaunay Triangulation [48] to triangulate the polygons in our dataset, ensuring a unique triangulation for each polygon [49]. To increase data diversity, we perform rotation-based augmentation to generate multiple samples per triangulation graph. Rotation, rather than shear augmentation, is used because the triangulation graph structure changes with perturbations in vertex locations.

We evaluate the performance on the triangulation reconstruction problem by computing the Euclidean distance between corresponding vertices, as each triangulation is uniquely determined by the spatial configuration of the points. To account for rotation variations, all polygons are rotated such that their first edge is aligned with the x-axis. Table 1 presents the quantitative results of VisDiff compared to the baselines on the triangulation graph reconstruction problem. VisDiff performs comparably to MeshAnything and outperforms Vertex-Diffusion, GNN, and Conditional-VAE in terms of Euclidean distance. Since MeshAnything is trained using a triangulation-based representation, its strong performance on triangulation structures is expected. However, it is important to note that MeshAnything frequently produces disconnected triangle sets, whereas our approach consistently generates a closed polygonal chain. Metrics for MeshAnything are reported only for polygons forming a closed polygonal chain. Qualitative results are provided in Appendix D.4.

6.6 Polygon Sampling

In previous experiments, we demonstrated that VisDiff can learn to generate multiple polygons conditioned on the visibility and triangulation graphs. In this section, we leverage these capabilities to highlight the high-diversity sampling ability of VisDiff over the space of all polygons. Specifically, we apply VisDiff to generate valid interpolations between pairs of polygons that share the same visibility graph. Beyond polygon-to-polygon interpolation, we also introduce a graph-to-graph interpolation approach to sample diverse polygons across the polygon manifold.

Polygon-to-Polygon Interpolation: We perform polygon-to-polygon interpolation by sampling two different diffusion seeds and linearly interpolating between them. VisDiff is then used to generate polygons corresponding to the interpolated noise samples while keeping the visibility graph

fixed. We perform 50 interpolation steps in total. Qualitative results are provided in Appendix G.1, showing six intermediate steps from the polygon-to-polygon interpolation process. VisDiff produces meaningful intermediate polygons across the interpolation sequence, indicating that it learns a smooth neighborhood structure within the latent space of a visibility graph.

Graph-to-Graph Interpolation: Sampling between two valid graphs through interpolation is a challenging problem, as intermediate steps must correspond to valid graphs for which polygons exist. Triangulation graphs possess an important property: any two valid triangulations can be transformed into one another through a sequence of local operations known as *edge flips*, with all intermediate triangulations remaining valid [50]. This sequence forms a *flip graph*, which has been shown to be connected for 2D point sets [51]. We therefore exploit the capability of VisDiff to sample using intermediate triangulation graphs, enabling graph-to-graph interpolation over valid polygonal structures.

We first select two distinct valid triangulation graphs from the test dataset and apply the edge-flip algorithm [50] to generate intermediate triangulations between them. These intermediate triangulation paths are then used as inputs to VisDiff to generate the corresponding polygons. Qualitative results in Appendix G.2 show that VisDiff produces smooth interpolations between triangulation graphs. This observation further indicates that VisDiff learns a continuous manifold representation for triangulation graphs.

7 Applications

VisDiff enables several practical applications. The proposed dataset serves as a navigation benchmark for evaluating motion planners in high-occlusion settings. An example of this application is presented in [52]. Its polygon sampling capability supports data augmentation by generating diverse, valid polygons conditioned on the underlying graph structure. Finally, the insights from VisDiff can be integrated into floorplan generative models [53, 54] to incorporate privacy constraints in generated layouts.

8 Limitations and Future Work

At a high level, our results demonstrate that modern neural representations can encode the space of all polygons such that distances on the learned manifold remain faithful to their combinatorial properties. However, the current VisDiff architecture represents the SDF as a grid, leading to computational and memory bottlenecks as polygons with more vertices require finer grid resolutions. We plan to adopt more efficient SDF encodings, such as [55, 56]. Another limitation of VisDiff is that it does not achieve perfect visibility graph reconstruction accuracy. To address this, we aim to explore LLM-based coding approaches [57, 58] to incorporate additional consistency mechanisms.

9 Conclusion

In this paper, we studied the problems of reconstruction and characterization of simple polygons based on combinatorial graphs that lack an underlying distance metric, features, or neighborhood structure. We introduced **VisDiff**, a diffusion-based approach that first predicts the Signed Distance Function (SDF) associated with the input visibility graph G. The SDF is then used to generate the vertex locations of a polygon P whose visibility graph corresponds to G. We showed that incorporating the SDF leads to a 26% improvement in F1-score compared to state-of-the-art approaches on the *Visibility Reconstruction* problem. We further demonstrated the ability of VisDiff to sample multiple polygons for a single visibility graph G, addressing the *Visibility Characterization* problem. Additionally, we presented preliminary results achieving 90% classification accuracy on the *Visibility Recognition* problem. Beyond visibility graphs, VisDiff also generalizes to triangulation as inputs. Finally, we leveraged these capabilities to perform diverse polygon sampling through both graph-to-graph and polygon-to-polygon interpolation.

Acknowledgment. This work was funded in part by the National Research Foundation of Korea (NRF) grant (MSIT) No. RS-2024-00462874.

References

[1] Paul A Longley, Michael F Goodchild, David J Maguire, and David W Rhind. *Geographic information science and systems*. John Wiley & Sons, 2015.

[2] Keundeok Park, Semiha Ergan, and Chen Feng. Quality assessment of residential layout designs generated by relational generative adversarial networks (gans). *Automation in Construction*, 158:105243, 2024.

[3] Peter Werner, Alexandre Amice, Tobia Marcucci, Daniela Rus, and Russ Tedrake. Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10359–10365. IEEE, 2024.

[4] George Nagy. Terrain visibility. *Computers & graphics*, 18(6):763–773, 1994.

[5] Yuanwen Yue, Theodora Kontogianni, Konrad Schindler, and Francis Engelmann. Connecting the dots: Floorplan reconstruction using two-level queries, 2023.

[6] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3DGen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

[7] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Hang Yu, Wei Liu, Xiangyang Xue, and Yu-Gang Jiang. Pixel2Mesh: 3D mesh model generation via image guided deformation. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3600–3613, 2020.

[8] Zian Li, Xiyuan Wang, Yinan Huang, and Muhan Zhang. Is distance matrix enough for geometric deep learning? *Advances in Neural Information Processing Systems*, 36, 2024.

[9] 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.

[10] 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.

[11] Safwa Ameer, Matt Gibson-Lopez, Erik Krohn, and Qing Wang. On the visibility graphs of pseudo-polygons: recognition and reconstruction. In *18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

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

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

[14] 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*, pages 72–89. Springer, 2016.

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

[16] Hazel Everett. *Visibility graph recognition*. University of Toronto, 1990.

[17] Hossein Boomari, Mojtaba Ostovari, and Alireza Zarei. Recognizing visibility graphs of polygons with holes and internal-external visibility graphs of polygons. *arXiv preprint arXiv:1804.05105*, 2018.

[18] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-SDF: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2262–2272, 2023.

[19] Jiacheng Chen, Ruizhi Deng, and Yasutaka Furukawa. PolyDiffuse: Polygonal shape reconstruction via guided set diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. SDFusion: Multimodal 3D shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023.

[21] Jaehyeok Shim, Changwoo Kang, and Kyungdon Joo. Diffusion-based signed distance fields for 3D shape generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20887–20897, 2023.

[22] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19615–19625, 2024.

[23] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. PolyDiff: Generating 3D polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023.

[24] Guanyu Cui and Zhewei Wei. MGNN: Graph neural networks inspired by distance geometry problem. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 335–347, 2023.

[25] Dazhou Yu, Yuntong Hu, Yun Li, and Liang Zhao. PolygonGNN: Representation Learning for Polygonal Geometries with Heterogeneous Visibility Graph. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4012–4022, 2024.

[26] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents, 2022.

[27] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

[28] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[29] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. *arXiv preprint arXiv:2010.02502*, 2020.

[31] Frank Permenter and Chenyang Yuan. Interpreting and Improving Diffusion Models from an Optimization Perspective. *arXiv preprint arXiv:2306.04848*, 2023.

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[34] Khoa Anh Ngo, Kyuhong Shim, and Byonghyo Shim. Spatial Cross-Attention for Transformer-Based Image Captioning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[35] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation, 2021.

[36] Jiang Liu, Hui Ding, Zhaowei Cai, Yuting Zhang, Ravi Kumar Satzoda, Vijay Mahadevan, and R. Manmatha. Polyformer: Referring image segmentation as sequential polygon generation, 2023.

[37] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9131–9140, 2020.

[38] Yiming Xie, Huaizu Jiang, Georgia Gkioxari, and Julian Straub. Pixel-aligned recurrent queries for multi-view 3d object detection, 2023.

[39] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization, 2019.

[40] Jiang Liu, Hui Ding, Zhaowei Cai, Yuting Zhang, Ravi Kumar Satzoda, Vijay Mahadevan, and R Manmatha. Polyformer: Referring image segmentation as sequential polygon generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18653–18663, 2023.

[41] Maheswari Visvalingam and James D Whyatt. Line generalization by repeated elimination of points. In *Landmarks in Mapping*, pages 144–155. Routledge, 2017.

[42] Xiaoyao Ding, Shaopeng Duan, and Zheng Zhang. Semantic-guided attention and adaptive gating for document-level relation extraction. *Scientific Reports*, 14(1):26628, 2024.

[43] Thomas Auer and Martin Held. Heuristics for the generation of random polygons. In *CCCG*, pages 38–43, 1996.

[44] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models, 2023.

[45] William Harvey, Saeid Naderiparizi, and Frank Wood. Conditional image generation by conditioning variational auto-encoders, 2022.

[46] Guanyu Cui and Zhewei Wei. Mgnn: Graph neural networks inspired by distance geometry problem, 2023.

[47] Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[48] L Rognant, Jean-Marc Chassery, S Goze, and JG Planes. The Delaunay constrained triangulation: the Delaunay stable algorithms. In *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*, pages 147–152. IEEE, 1999.

[49] Simena Dinas and José María Banon. A review on Delaunay triangulation with application on computer vision. *Int. J. Comput. Sci. Eng*, 3:9–18, 2014.

[50] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. In *Proceedings of the twelfth annual symposium on Computational geometry*, pages 214–223, 1996.

[51] Charles L Lawson. Transforming triangulations. *Discrete mathematics*, 3(4):365–372, 1972.

[52] Yukang Cao, Rahul Moorthy, O. Goktug Poyrazoglu, and Volkan Isler. C-Free-Uniform: A Map-Conditioned Trajectory Sampler for Model Predictive Path Integral Control, 2025.

[53] Shibo Hong, Xuhong Zhang, Tianyu Du, Sheng Cheng, Xun Wang, and Jianwei Yin. Cons2Plan: Vector Floorplan Generation from Various Conditions via a Learning Framework based on Conditional Diffusion Models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3248–3256, 2024.

[54] Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising, 2022.

[55] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[56] Eric Mitchell, Selim Engin, Volkan Isler, and Daniel D Lee. Higher-Order Function Networks for Learning Composable 3D Object Representations. In *International Conference on Learning Representations*, 2020.

[57] Ansh Nagda, Prabhakar Raghavan, and Abhradeep Thakurta. Reinforced Generation of Combinatorial Structures: Applications to Complexity Theory, 2025.

[58] Carlo Bosio, Matteo Guarrera, Alberto Sangiovanni-Vincentelli, and Mark W. Mueller. Combining Large Language Models and Gradient-Free Optimization for Automatic Control Policy Synthesis, 2025.

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

[60] 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*, pages 740–755. Springer, 2014.

[61] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and Introduction both focus on current approaches failing to capture combinatorial structures without underlying distance metrics and proposed an approach **VisDiff** to solve it.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Stated it in Section 8

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our work focuses on current approaches failing to capture combinatorial structures without underlying distance metrics. We present **VisDiff** which uses SDF as intermediate representation to solve it.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide architecture and training details in Appendix section J, Evaluation and approach details in Section 6 and 4

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide open access to data and code which has been added as link in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide architecture and training details in Appendix section J. Additionally all evaluation details in Section 6

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We show the diversity and difference between the training and testing dataset in Appendix Section B

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the training details in Appendix Section J

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We uphold the code of ethics of Neurips.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work does not have societal impact

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We don't use any scraped dataset for our work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the papers of the baselines used to compare our approach.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release a new dataset which has been added as a link to the paper and the link also provides documentation on how to use it.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not do any crowdsourcing experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No IRB approval required for our work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: No LLM used for the work.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

We first provide formal definitions of the terms used in this paper in Section A. Next, we describe the dataset generation process in Section B. The ablation study analyzing the design choices of our method is presented in Section C. We then provide additional results for *Visibility Reconstruction* (Section D), *Visibility Characterization* (Section E), and *Visibility Recognition* (Section F). In Section G, we present further results demonstrating the ability of VisDiff to sample polygons on a continuous polygon manifold. We also evaluate intermediate results of the SDF diffusion process in Section H and analyze the effect of SDF error on the vertex generation block in Section I. Finally, we provide model hyperparameters, architectural details, and training configurations in Section J.

A Definitions

We provide more formal definitions of the terms simple polygon and visibility graph in Table 3.

Terms	**Definitions**
Simple Polygon	Let $V = (v_1, \ldots, v_n)$ be an ordered set of n points on the plane. The location of point v_i is specified by its coordinates (x_i, y_i). Let $e_i = (v_i, v_{i+1})$ be the set of line segments obtained by connecting consecutive points in V in a cyclic manner. These line segments define a closed planar curve – the boundary of a polygon P. The points v_i are the *vertices* of P and the segments e_i are its *sides*. Two consecutive edges of a polygon share an end-point at a vertex. In a simple polygon, these are the only intersections between the edges. The edges do not intersect each other.
Visibility Graph	A simple polygon P has a well-defined interior and an exterior separated by its boundary δP. This separation allows us to define *visibility*: We will use the notation $x \in P$ to denote that x lies either on the boundary or the interior of P. We say that two points $x, y \in P$ *see each other* if and only if $\forall z \in [x, y], z \in P$. In other words, the line segment $[xy]$ lies completely inside or on the boundary of P. The visibility graph of P, denoted $G(P)$, is a graph that is a vertex-to-vertex relation of P. There is an edge between two vertices u and v if and only if u and v are visible to each other in P.

Table 3: Definitions

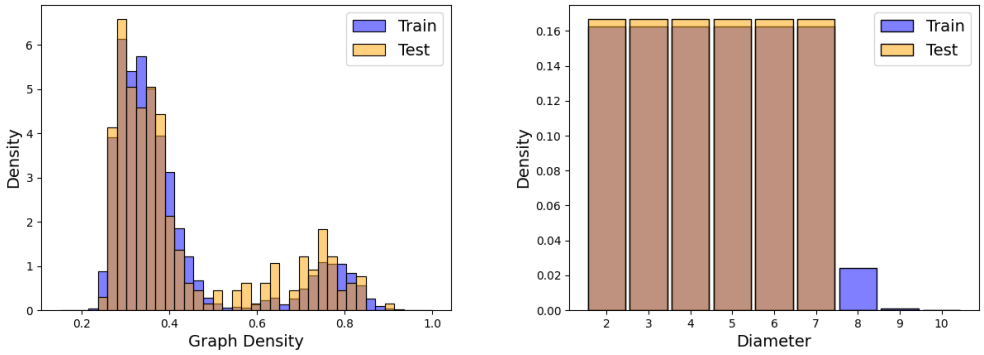
B Dataset Generation

B.1 Dataset Statistics

In this section, we present detailed statistics of our dataset. Figure 5 shows the distribution of the training and in-distribution test set, indicating that our dataset is uniform with respect to the link diameter of the visibility graph. Figure 6 compares the training dataset with the out-of-distribution test set, showing that the star, convex-fan, and terrain classes have edge densities that differ from the training distribution, where density refers to the ratio of visible edges to total possible edges in the visibility graph. Table 4 provides a comparison between the VisDiff dataset and other real-world datasets in terms of visibility graph diversity based on link diameter. The results demonstrate that the VisDiff dataset exhibits significantly greater diversity than existing real-world datasets.

B.2 Test Set Generation

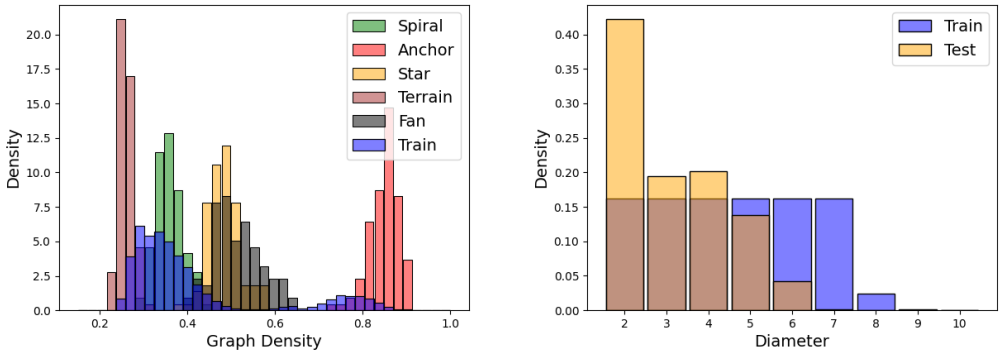
We generate two datasets for evaluation: in-distribution and out-of-distribution. In-distribution samples are generated by setting aside 100 unique polygons per link diameter from the large dataset. These are not included in the training data.



(a) Density Comparison Train vs Test

(b) Diameter Comparison Train vs Test

Figure 5: Train vs in-distribution test set analysis: 5a) The density is inversely proportional to the diameter. Uniform sampling of diameter results in bimodal density. 5b) Training and testing sets are uniform in terms of the link diameter of the visibility graph.



(a) Density Comparison Train vs Test

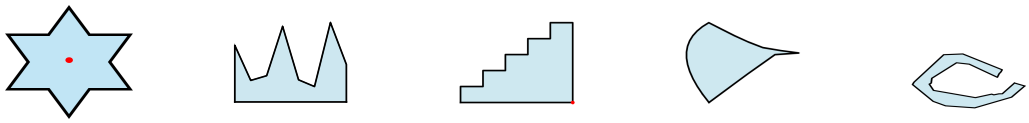
(b) Diameter Comparison Train vs Test

Figure 6: Out-of-distribution test set analysis: Figure 6a shows the density of the anchor and spiral are close to the mean of the bimodal training distribution, making it similar to our training set. The density of the star, convex fan, and terrain differ significantly from the training distribution.

Baselines	Link Diameter (Avg / Std)	Min	Max
MNIST [59]	1.83 / 0.50	1.0	4.1
COCO 2017 [60]	1.32 / 0.25	1.0	3.823
VisDiff	**4.4 / 2.2**	1.0	**9.0**

Table 4: Real-world dataset comparison showing the diversity (in link diameter) between the VisDiff dataset and standard benchmarks.

The out-of-distribution samples are generated based on specific polygon types: star, spiral, anchor, convex-fan, and terrain. Figure 7 illustrates the properties of these polygon types. The spiral and anchor polygons share characteristics similar to those in our dataset, whereas the terrain, convex-fan, and star polygons differ significantly in terms of density, defined as the ratio of visible edges to total possible edges in the visibility graph. Figure 6 highlights the differences in visibility graph density for the terrain, convex-fan, and star classes compared to the training set.



(a) Star (b) Terrain (c) Fan (d) Anchor (e) Spiral

Figure 7: Polygon types: a) **Star**: Single kernel point (red) from which all vertex locations are visible, b) **Terrain**: X-monotone polygons where orthogonal lines from the X axis intersect the polygon boundary at most twice, c) **Convex Fan**: Single convex vertex (red) which appears in every triangle of the polygon triangulation, d) **Anchor**: Polygons with two reflex links and a convex link connecting both of them, e) **Spiral**: Polygons with long link diameter.

C Ablation Study

In this section, we present a performance comparison of our architecture using different feature types. Specifically, we analyze the contribution of each feature type to visibility reconstruction performance through an ablation study. Table 5 summarizes the results of this comparison.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) Global	0.894	0.884	0.873	0.876
(b) Pixel	0.869	0.859	0.839	0.843
(c) VisDiff	**0.924**	**0.914**	**0.911**	**0.912**

Table 5: Ablation comparison: (a) Global patch-based features, (b) Pixel-aligned local features, and (c) Combined global patch-based + pixel-aligned local features. **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall.

D Visibility Reconstruction Results

In this section, we present a quantitative comparison of VisDiff with baselines on the out-of-distribution dataset for the *Visibility Reconstruction* problem. We also provide additional qualitative results for the *Visibility Reconstruction* problem.

D.1 Out-of-distribution Comparison

In this section, we present the baseline performance on the out-of-distribution dataset for the *Visibility Reconstruction* problem. Table 6 compares VisDiff with baseline methods on this dataset. The F1-score comparison shows that VisDiff performs significantly better than all baselines on the out-of-distribution dataset.

	Acc ↑	Prec ↑	Rec ↑	F1 ↑
(a) VD	0.751	0.734	0.702	0.697
(b) C-VAE	0.733	0.713	0.699	0.694
(c) GNN	0.666	0.732	0.643	0.610
(d) Ours	**0.915**	**0.895**	**0.891**	**0.891**
(e) Mesh	0.708	0.715	0.709	0.675

Table 6: Baseline comparison: (a) Vertex-Diffusion, (b) Conditional-VAE, (c) GNN, (d) VisDiff, and (e) MeshAnything. **Acc**: Accuracy, **Prec**: Precision, **Rec**: Recall.

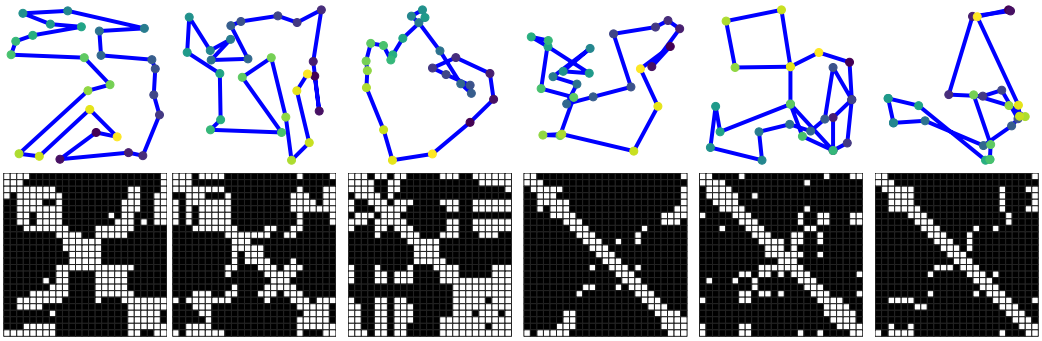
D.2 Reconstruction Generation Strategy

In this section, we describe the approach used in VisDiff for polygon generation and selection in the *Visibility Reconstruction* problem. We leverage the capability of VisDiff to generate multiple polygon

samples for a given visibility graph. The visibility graph of each generated polygon is computed, and the final polygon is selected based on the highest F1-score with respect to the target visibility graph. We use 50 samples, chosen as a trade-off between performance and computational cost across different sample sizes. For a fair comparison, we apply the same generation and selection procedure to Vertex Diffusion and C-VAE, since both methods are also capable of generating multiple polygons for a given visibility graph.

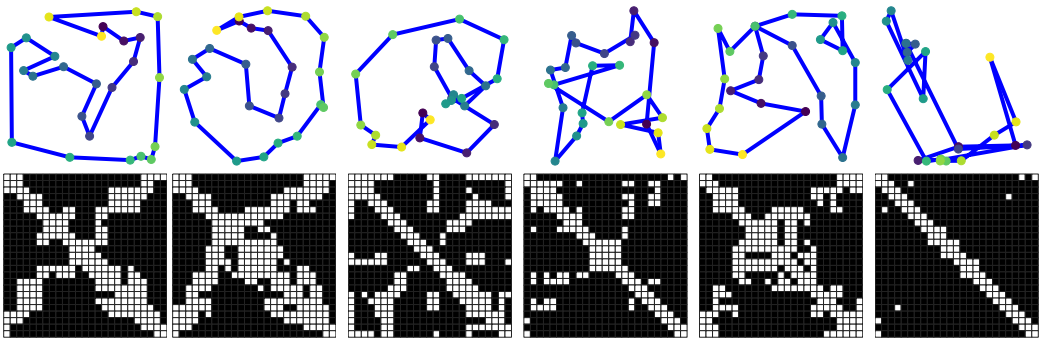
D.3 Qualitative Results

We provide additional qualitative results for the *Visibility Reconstruction* problem. Figures 8, 9 and 10 show the comparison between polygons generated by VisDiff to baselines. The F1-Score shows that VisDiff generates polygons much closer to the visibility graph of the ground truth polygon.



(a) GT (b) Ours: **0.855** (c) C-VAE: 0.653 (d) VD: 0.736 (e) Mesh: 0.745 (f) GNN: 0.76

Figure 8: *Visibility Reconstruction*: The top row shows the polygons generated by different methods. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows corresponding visibility graphs of the polygons where **white** represents the visible edge and **black** represents the non-visible edge. The captions indicate the F1 Score of the visibility graph compared to the GT.

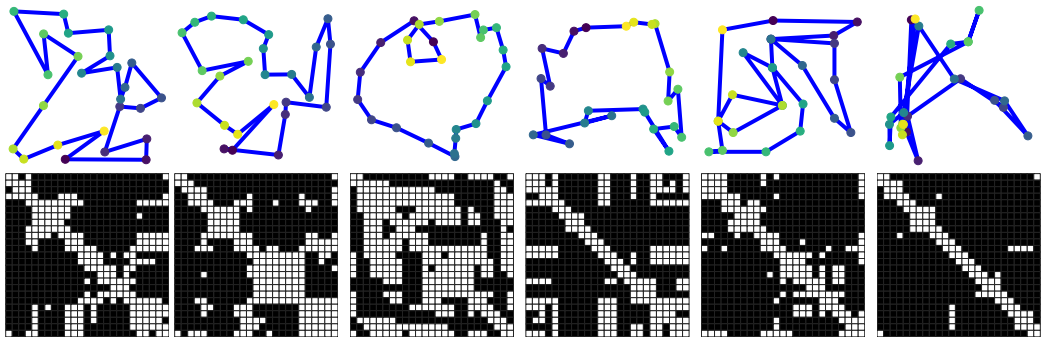


(a) GT (b) Ours: **0.846** (c) C-VAE: 0.65 (d) VD: 0.765 (e) Mesh: 0.759 (f) GNN: 0.706

Figure 9: Visibility reconstruction qualitative results: The top row shows the polygons generated by different methods. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows corresponding visibility graphs of the polygons where **white** represents the visible edge and **black** represents the non-visible edge. The captions indicate the F1 Score of the visibility graph compared to the GT.

D.4 Triangulation Results

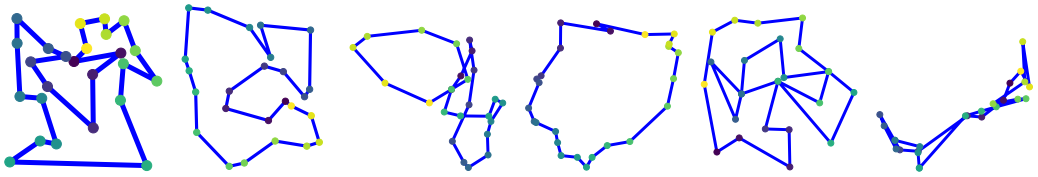
In addition to conditioning on the visibility graph, we also provide qualitative results for the problem of generating polygons from triangulation graphs. Figures 11 and 12 show the performance of VisDiff compared to other baselines. It can be observed that both MeshAnything and VisDiff generate



(a) GT (b) Ours: **0.85** (c) C-VAE: 0.594 (d) VD: 0.71 (e) Mesh: 0.765 (f) GNN: 0.776

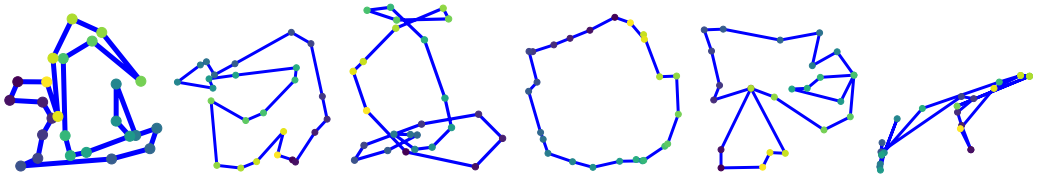
Figure 10: Visibility reconstruction qualitative results: The top row shows the polygons generated by different methods. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows corresponding visibility graphs of the polygons where **white** represents the visible edge and **black** represents the non-visible edge. The captions indicate the F1 Score of the visibility graph compared to the GT.

polygons close to the ground truth, whereas Vertex Diffusion, C-VAE, and GNN fail to accurately interpret the triangulation graph.



(a) GT (b) Ours (c) C-VAE (d) VD (e) Mesh (f) GNN

Figure 11: Triangulation Qualitative Results: Top row shows the polygons generated by different methods. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering).



(a) GT (b) Ours (c) C-VAE (d) VD (e) Mesh (f) GNN

Figure 12: Triangulation Qualitative Results: Top row shows the polygons generated by different methods. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering).

D.5 Computational Cost Comparison

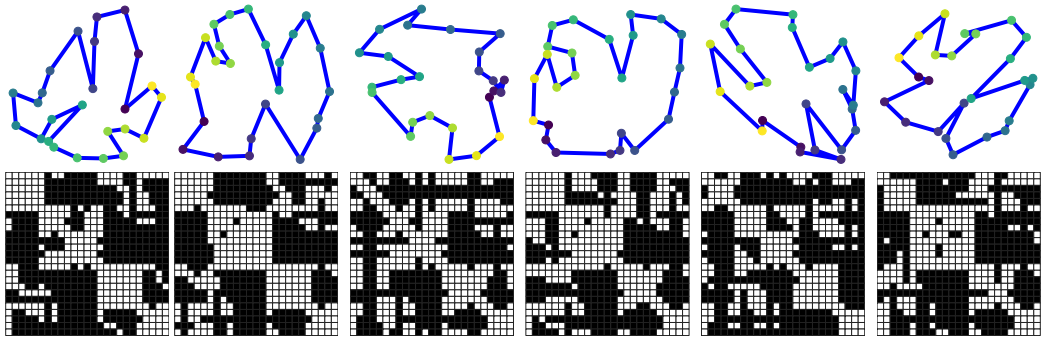
In Table 7, we compare the computational cost of our model with the baselines by measuring the inference time per sample. The inference time of VisDiff is higher than that of the baseline models, as VisDiff performs inference in two stages through the SDF, whereas the baselines complete it in a single step.

Baselines	Computational Time (seconds) ↓
Mesh	0.40
C-VAE	0.003
GNN	0.005
VD	0.094
Ours	1.02

Table 7: Computational cost comparison: Inference time (in seconds) required by each model to generate vertex locations for a single visibility graph.

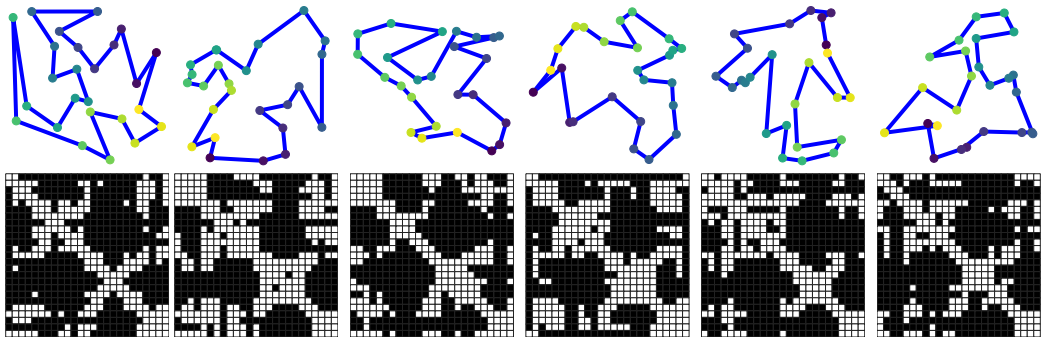
E Visibility Characterization Results

We provide additional qualitative results for the *Visibility Characterization* problem. Figures 13 and 14 demonstrate the ability of VisDiff to sample multiple polygons given the same visibility graph.



(a) GT (b) 0.802 (c) 0.77 (d) 0.765 (e) 0.8 (f) 0.766

Figure 13: *Visibility Characterization*: The top row shows multiple polygons generated by VisDiff for the same visibility graph G. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows the visibility graph corresponding to the polygons where **white** represents visible edge and **black** represents non-visible edge. The caption shows the F1-Score compared to the ground truth (GT) visibility graph.

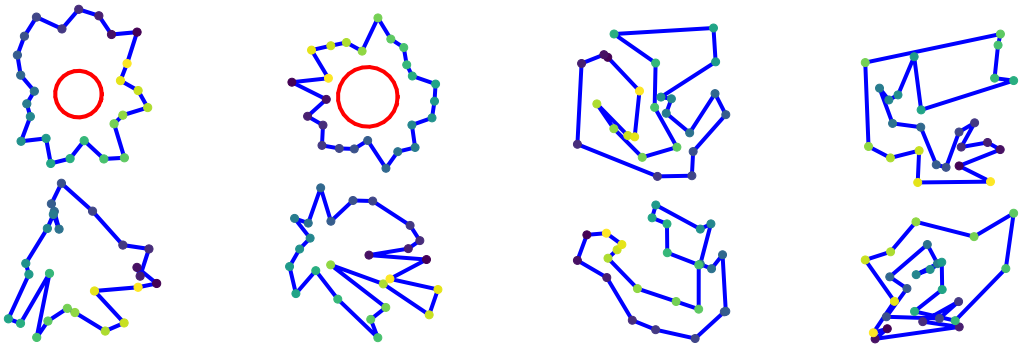


(a) GT (b) 0.782 (c) 0.78 (d) 0.8 (e) 0.752 (f) 0.762

Figure 14: *Visibility Characterization*: The top row shows multiple polygons generated by VisDiff for the same visibility graph G. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The second row shows the visibility graph corresponding to the polygons where **white** represents visible edge and **black** represents non-visible edge. The caption shows the F1-Score compared to the ground truth (GT) visibility graph.

F Visibility Recognition Results

We provide additional qualitative results illustrating both successful and failure cases of VisDiff on the *Visibility Recognition* problem, where the input may be a valid or non-valid visibility graph. Invalid samples are generated by constructing visibility graphs of polygons with holes. Figures 15a and 15c show successful reconstructions for valid and non-valid visibility graphs, respectively, while Figures 15b and 15d illustrate failure cases. These results demonstrate that VisDiff can identify non-valid visibility graphs in most scenarios when used as a classifier based on the validity of its generated outputs.



(a) (b) (c) (d)

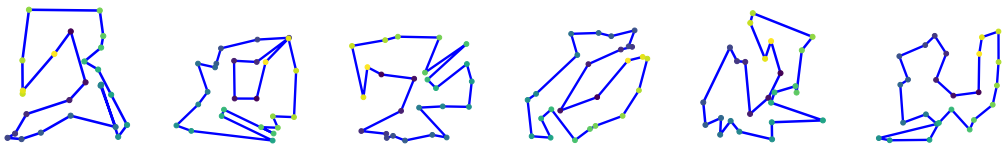
Figure 15: *Visibility Recognition*: The top row signifies the ground truth non-valid polygon with the hole (red) while the bottom row is the polygons drawn by VisDiff. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). a) Non-Valid Sample 1: VisDiff predicts it as a non-valid polygon as it is not able to generate any valid polygon, b) Non-Valid Sample 2: VisDiff generates valid polygon where it learns to put points in a V shape to account for a hole. It misclassified a non-valid visibility graph as a valid visibility graph. c) Valid Sample 1: VisDiff predicts it as a non-valid polygon as it is not able to generate any valid polygon, d) Valid Sample 2: VisDiff predicts it as a valid visibility graph as it is able to generate any valid polygon with high F1 relative to target visibility graph

G Polygon Sampling

We provide qualitative results demonstrating the capability of VisDiff to perform high-diversity data sampling within the polygon manifold space. We present qualitative examples for both the polygon-to-polygon interpolation and graph-to-graph interpolation approaches.

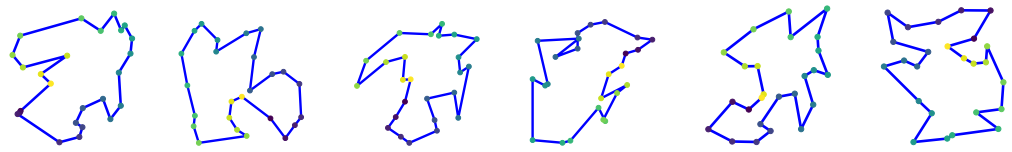
G.1 Polygon-to-Polygon Interpolation

We provide qualitative results for the polygon-to-polygon interpolation approach. Figures 16 and 17 visualize six interpolation steps for two polygon samples. VisDiff generates meaningful intermediate polygons across the interpolation sequence.



(a) $T = 0$ (b) $T = 10$ (c) $T = 20$ (d) $T = 30$ (e) $T = 40$ (f) $T = 50$

Figure 16: Polygon-to-Polygon Interpolation: The top row shows different polygons generated by VisDiff at different instances of 50 steps during linear interpolating between two noise samples the same visibility graph G. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The caption shows the timestep of interpolation between the two samples.

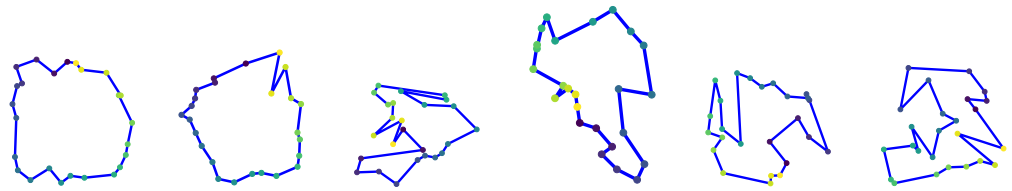


(a) $T = 0$ (b) $T = 10$ (c) $T = 20$ (d) $T = 30$ (e) $T = 40$ (f) $T = 50$

Figure 17: Polygon-to-Polygon Interpolation: The top row shows different polygons generated by VisDiff at different instances of 50 steps during linear interpolating between two noise samples the same visibility graph G. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The caption shows the timestep of interpolation between the two samples.

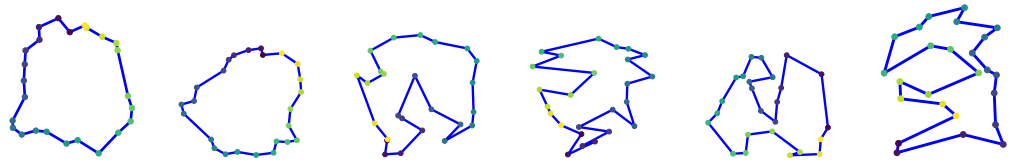
G.2 Graph-to-Graph Interpolation

We provide qualitative results for the graph-to-graph interpolation approach. Figures 18 and 19 visualize six randomly sampled intermediate interpolation steps for two triangulation graph examples. VisDiff generates meaningful intermediate polygons between the triangulation graphs of convex and concave polygons.



(a) (b) (c) (d) (e) (f)

Figure 18: Graph-to-Graph Interpolation: The top row shows different polygons generated by VisDiff at different interpolation instances of two triangulation graphs. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The caption shows the timestep of interpolation between the two samples.



(a) (b) (c) (d) (e) (f)

Figure 19: Graph-to-Graph Interpolation: The top row shows different polygons generated by VisDiff at different interpolation instances of two triangulation graphs. The first vertex is represented by deep purple and the last vertex by yellow (anticlockwise ordering). The caption shows the timestep of interpolation between the two samples.

H SDF Diffusion Evaluation

We evaluate the SDF diffusion model by measuring the L2 error between the ground-truth and predicted SDFs on both in-distribution and out-of-distribution test datasets for the *Visibility Reconstruction* problem. Table 8 presents the performance of the SDF diffusion model. The results show that our diffusion model predicts high-quality SDFs with low L2 error, demonstrating its effectiveness in capturing the underlying relationship between polygons and their visibility graphs.

Test Dataset	L2 Error ↓
In-Distribution	0.071
Out-Distribution: Spiral	0.091
Out-Distribution: Terrain	0.091
Out-Distribution: Convex Fan	0.083
Out-Distribution: Anchor	0.158
Out-Distribution: Star	0.069

Table 8: SDF evaluation: L2 error between the predicted SDF from the diffusion model and the ground-truth SDF across in- and out-distribution test sets.

I Vertex Prediction Analysis

In this section, we present experiments analyzing the effect of SDF generation on the vertex prediction block. We also provide a quantitative comparison between traditional polygon extraction methods and our vertex prediction block.

I.1 SDF-to-Polygon Error Analysis

We analyze the impact of perturbations in the SDF on the downstream vertex extraction block. Specifically, we report changes in performance metrics when varying levels of noise are added to the SDF input. We also evaluate the effect on the vertex prediction block when the visibility conditioning is removed during SDF generation. Table 9 summarizes the performance of the vertex prediction block across both experiments. As the SDF becomes increasingly distorted, the accuracy of vertex extraction consistently degrades, since the zero-level set becomes harder to localize and fails to form a well-defined polygon consistent with the visibility graph. We also observe a noticeable drop in performance compared to conditional SDF and polygon generation, indicating that reconstruction quality strongly depends on the fidelity of the conditional SDF.

SDF Noise Standard Deviation	F1 Score ↑
0.0 (Conditional SDF)	**0.912**
0.0 (Unconditional SDF)	0.865
0.01	**0.912**
0.05	0.909
0.1	0.905
0.5	0.859

Table 9: SDF-to-polygon error analysis: F1 score between the visibility graph of the generated polygon (from the vertex prediction block) and the ground-truth polygon under varying SDF noise levels. Conditional and unconditional SDF generation results are also compared.

I.2 Vertex Prediction Baseline Comparison

We compare our proposed vertex prediction block with the standard polygon extraction approach, specifically the Marching Cubes algorithm. Table 10 presents a comparison between VisDiff and Marching Cubes. VisDiff significantly outperforms Marching Cubes across all evaluation metrics.

Approach	F1 Score ↑
VisDiff	**0.912**
Marching Cubes Algorithm [61]	0.800

Table 10: Vertex prediction baseline comparison: F1 score between the visibility graph of the polygon generated by the vertex prediction block and the ground-truth polygon, compared against a traditional polygon extraction approach (Marching Cubes).

J Training Details

In this section, we present the training setup and detailed architecture of **VisDiff**. We also provide a quantitative comparison with the joint training variant of VisDiff. All models were trained on a single NVIDIA A100 GPU using 10 workers, with a total training time of approximately 16 hours.

J.1 SDF Diffusion

The SDF diffusion block uses a time-conditioned U-Net architecture with three downsampling layers having channel sizes of 32, 64, and 128. Each downsampling block is followed by a Spatial Transformer layer, which performs cross-attention with the visibility features. The bottleneck feature consists of 512 channels and is followed by upsampling layers with channel sizes of 128, 64, 32, and finally 1. Skip connections are maintained between the encoder and decoder, consistent with the standard U-Net structure. Additionally, a Spatial Transformer layer is applied after each upsampling block, and all U-Net layers use ReLU activation functions.

We train the SDF diffusion model for 60 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128. We employ a log-linear noise scheduler with $\sigma_{\text{min}} = 0.005$ and $\sigma_{\text{max}} = 10$.

J.2 Vertex Extraction Block

The vertex extraction block consists of two modules: the SDF encoding module and the vertex prediction block. Both modules were trained for 60 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128. We now describe the architecture and training setup used for these modules.

J.2.1 SDF Encoding

The SDF encoding block uses a U-Net architecture to extract pixel-aligned features. The encoder consists of convolutional layers with channel sizes of 64, 128, 256, and 512, while the decoder contains upsampling layers with channel sizes of 256, 128, 64, and 128, producing the final pixel-aligned feature maps. All layers use ReLU activation functions. The bottleneck feature is represented as $Z_{\text{global}} \in \mathbb{R}^{5\times5\times512}$, which is flattened to $\mathbb{R}^{25\times512}$. Positional embeddings are added to these flattened patches to preserve spatial ordering.

J.2.2 Vertex Prediction Block

The vertex prediction block consists of three transformer encoder layers, each with 256 hidden units. The contour initialized from the SDF is provided as input to the transformer, and the output passes through a multilayer perceptron (MLP) with layers of 256 and 2 units to predict the final (x, y) coordinates for each polygon query. Figure 20 illustrates the detailed architecture of the transformer block.

J.3 Joint vs. Two-Stage Training

We compare the performance of our two-stage training approach with joint training of the SDF diffusion and vertex prediction blocks without vertex initialization. Table 11 presents the results of this comparison. Our two-stage training with vertex initialization significantly outperforms the joint training approach.

Training Approach	F1 Score ↑
Joint Training	0.850
Two-Stage Training with Vertex Initialization	**0.912**

Table 11: Training approach comparison: F1 score between the visibility graph of the polygon generated by the vertex prediction block and the ground-truth polygon, comparing joint training and two-stage training with vertex initialization.

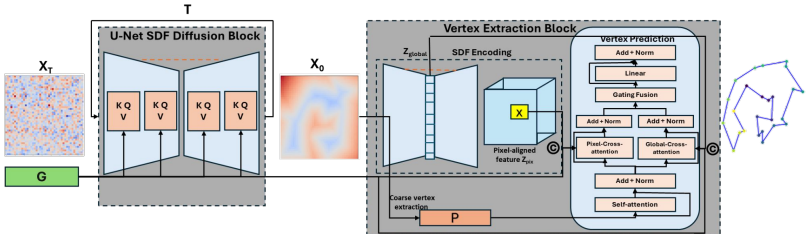


Figure 20: VisDiff architecture. The model consists of two main components: the U-Net SDF Diffusion block and the Vertex Extraction block. **U-Net Diffusion Block:** A noisy SDF, denoted as $\mathbf{X}_T$, is first sampled from a Gaussian distribution. $\mathbf{X}_T$ then passes through $\mathbf{T}$ timesteps of the reverse diffusion process to produce the clean SDF $\mathbf{X}_0$. This denoising process is conditioned on the input graph $\mathbf{G}$ using transformer cross-attention blocks represented by $\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$, which correspond to the key, query, and value terms, respectively. In our approach, $\mathbf{Q}$ is obtained from the learned spatial CNN features, while $\mathbf{K}$ and $\mathbf{V}$ are derived from $\mathbf{G}$. An initial set of vertices $\mathbf{P}$ is then estimated from $\mathbf{X}_0$ through contour extraction. **Vertex Extraction Block:** Given the predicted SDF $\mathbf{X}_0$, the SDF encoder generates pixel-aligned features Z_{pix} and global features Z_{global}. These features, together with the initial vertices $\mathbf{P}$, are passed to the vertex prediction block to estimate the final vertex locations. During **Training**, the model is supervised using both the ground-truth SDF and polygon. During **Testing**, only the visibility graph $\mathbf{G}$ is provided as input.