# PI3DETR: Parametric Instance Detection of 3D Point Cloud Edges With a Geometry-Aware 3DETR

Supplementary Material

## A. Dataset

**Dataset filtering.** To ensure consistent supervision and stable evaluation, we apply a filtering protocol to the ABC dataset [9]. The dataset is organized in chunks of 10,000 models, and we use the first five chunks. Following [23], which reports that over 95% of curves are lines, circles, or B-splines and discards models with more than 300.000 vertices, we restrict our study accordingly. We add arcs as a separate class to capture non-closed circles and B-splines that are well represented by circular arcs, and we model remaining B-splines as cubic Bézier curves for better parameterization. We retain only models containing circles, cubic Bézier curves, line segments, or arcs. Models with more than 100 curves are removed. We also discard models containing circles with radius below 1% and lines shorter than 0.01% of the bounding box diagonal. Finally, we manually remove erroneous models, bevel heavy shapes, and models containing only circles. The resulting 10,240 samples are split into 80%for training, 10% for validation, and 10% for testing.

## B. PI3DETR Architecture

This section details PI3DETR's architecture, providing a complete explanation of the geometry-aware matching strategy, the computation of loss weights, the definitions of the loss functions, and the evaluation metrics.

### B.1. Geometry-Aware Matching

This section provides a slightly more detailed explanation of the matching procedure. To train the model, we need to match the set of predicted 3D curves to the ground truth curves. Since PI3DETR computes parameters for all four curve types for each decoder output embedding in $\{o_j^{\mathrm{f}}\}_{j=1}^{K}$, we eventually obtain $4K$ curves that are handled by the geometry-aware matcher. Given a prediction with index $j \in \{1,\dots,K\}$ and a ground truth curve with index $i \in \{1,\dots,M\}$, the matching cost $\mathcal{C}_{\text{match}}$ for a pair $(j,i)$ of curves is

$$\mathcal{C}_{\text{match}}(j,i) = -\hat{p}_j(c_i) + \mathcal{L}_{\text{param}}(j,i), \tag{6}$$

where the class cost $-\hat{p}_j(c_i)$ is the negative predicted probability of query $j$ being class $c_i$ [3], and $\mathcal{L}_{\text{param}}(j,i)$ is the geometric cost (Eq. (7)) selected accordingly to Eq. (2). We use cost and loss interchangeably for function reuse. Unlike the final loss in Eq. (5), however, the matcher does not compute gradients.

$$\begin{aligned}\mathcal{L}_{\text{param}}(j,i) = \\ \mathbb{1}_{\{c_i=1\}}\mathcal{L}_{\text{seq}}(\hat{\mathbf{B}}_j,\mathbf{B}_i) + \mathbb{1}_{\{c_i=2\}}\mathcal{L}_{\text{hybrid}}(\hat{\mathbf{L}}_j\mathbf{L}_i,)+ \\ \mathbb{1}_{\{c_i=3\}}\mathcal{L}_{\text{hybrid}}(\hat{\mathbf{C}}_j,\mathbf{C}_i,) + \mathbb{1}_{\{c_i=4\}}\mathcal{L}_{\text{seq}}(\hat{\mathbf{A}}_j,\mathbf{A}_i)\end{aligned} \tag{7}$$

The loss function $\mathcal{L}_{\text{seq}}$ in Eq. (3) measures the discrepancy between two curves parameterized as ordered point sequences (hence $\mathcal{L}_{\text{seq}}$, encompassing both cubic Bézier curves and arcs. Given a predicted sequence $\hat{\mathbf{B}}$ and ground truth sequence $\mathbf{B}$, the loss is defined as

$$\mathcal{L}_{\text{seq}}(\hat{\mathbf{B}},\mathbf{B}) = \min\{\|\hat{\mathbf{B}}-\mathbf{B}\|_1,\ \|\mathrm{rev}(\hat{\mathbf{B}})-\mathbf{B}\|_1\}, \tag{8}$$

and represents an order-invariant [7] $\ell_1$-loss. In Eq. (3), $\mathrm{rev}(\cdot)$ denotes sequence reversal. Specifically, for a cubic Bézier curve $\hat{\mathbf{B}} = [\hat{\mathbf{b}}^{\mathrm{s}},\hat{\mathbf{b}}^{\mathrm{c1}},\hat{\mathbf{b}}^{\mathrm{c2}},\hat{\mathbf{b}}^{\mathrm{e}}]$, we have $\mathrm{rev}(\hat{\mathbf{B}}) = [\hat{\mathbf{b}}^{\mathrm{e}},\hat{\mathbf{b}}^{\mathrm{c2}},\hat{\mathbf{b}}^{\mathrm{c1}},\hat{\mathbf{b}}^{\mathrm{s}}]$. For an arc $\hat{\mathbf{A}} = [\hat{\mathbf{a}}^{\mathrm{s}},\hat{\mathbf{a}}^{\mathrm{m}},\hat{\mathbf{a}}^{\mathrm{e}}]$, the reversed sequence is $\mathrm{rev}(\hat{\mathbf{A}}) = [\hat{\mathbf{a}}^{\mathrm{e}},\hat{\mathbf{a}}^{\mathrm{m}},\hat{\mathbf{a}}^{\mathrm{s}}]$. This formulation enforces invariance to the curve parameterization direction, reflecting the geometric equivalence of both orders.

We define a hybrid parameter loss $\mathcal{L}_{\text{hybrid}}$ in Eq. (4) to jointly handle line segments and circles. Both share parameters consisting of a 3D point, a direction or normal vector, and a scalar (length or radius). The loss combines an $\ell_1$-loss on the scalar with an $\ell_1$-loss on the point and vector pair. Since the direction vector is ambiguous up to sign, meaning $\mathbf{l}^{\mathrm{v}}$ and $-\mathbf{l}^{\mathrm{v}}$ represent the same orientation, the loss evaluates both options and uses the minimum to ensure invariance to vector direction, as shown in Fig. 3. Formally, the loss is defined as

$$\begin{aligned}\mathcal{L}_{\text{hybrid}}(\hat{\mathbf{L}},\mathbf{L}) = \min\{\|(\hat{\mathbf{l}}^{\mathrm{m}},\hat{\mathbf{l}}^{\mathrm{v}})-(\mathbf{l}^{\mathrm{m}},\mathbf{l}^{\mathrm{v}})\|_1, \\ \|(\hat{\mathbf{l}}^{\mathrm{m}},\hat{\mathbf{l}}^{\mathrm{v}})-(\mathbf{l}^{\mathrm{m}},-\mathbf{l}^{\mathrm{v}})\|_1\} + \|\hat{l}^{\mathrm{r}}-l^{\mathrm{r}}\|_1\,.\end{aligned} \tag{9}$$

This loss formulation applies analogously to a circle $\mathbf{C} = [\mathbf{c}^m,\mathbf{c}^v,c^r]$ parameterized by center point, unit normal vector, and radius. The sign ambiguity of the normal vector is handled identically, making $\mathcal{L}_{\text{hybrid}}$ a unified loss function for both lines and circles.

Based on the defined cost function, we compute a cost-minimal bipartite matching between the $K$ predictions and $M \le K$ ground truth curves using the Hungarian algorithm [10], following [3, 13]. The matching is represented by a permutation $\sigma \in \mathfrak{S}_K$ that assigns each ground truth index $i \in \{1,\dots,M\}$ to a unique prediction $j = \sigma(i)$. For the $K-M$ unmatched predictions, no-object class placeholders

with $c_{\sigma(i)} = 0$ are introduced and matched accordingly, extending $i$ to the full set $\{1, \ldots, K\}$ including these padded entries.

### B.2. Optional Post-Processing

Since PI3DETR directly regresses curves in 3D space, predictions may occasionally be slightly offset or duplicated. To mitigate this, we introduce two post-processing steps that are optional but can further enhance performance when properly parameterized as shown in the ablation study in Sec. 4.3. The first, Snap & Fit (S&F), aligns predicted curves with the underlying point cloud and enforces conformity to their geometric class (Fig. 4a). For each predicted curve $j$, points are sampled along the curve and matched to nearest neighbors in the point cloud, after which the curve is refit using a class-specific fitting method determined by the predicted class $\hat{c}_j = \arg\max_{k \in [0,4]} \hat{p}_j^{(k)}$. The second step, IoU Filter, removes duplicate or overlapping curves of the same class (Fig. 4b). Points are sampled along each curve, and pairwise distances between the sampled points of two curves are computed. Points within a predefined distance tolerance are considered overlapping, and the Intersection over Union (IoU) of these overlapping sets is calculated. If the IoU exceeds a threshold, the curve with lower confidence is discarded. In our experiments we set the IoU threshold to $0.6$ and the distance tolerance to $0.01$.

### B.3. Loss

**Class weight computation.** The class weights for the cross-entropy loss are computed from the distribution of curve classes in the training dataset. We use the class encoding $c = 0$ for the no-object class, $c = 1$ for cubic Bézier, $c = 2$ for line segment, $c = 3$ for circle, and $c = 4$ for arc. The sample counts for the non-empty classes are $n_1 = 11347$, $n_2 = 200751$, $n_3 = 34672$, and $n_4 = 37528$. The count for the no-object class is computed as $n_0 = K_{\text{total}} - (n_1 + n_2 + n_3 + n_4)$, where $K_{\text{total}} = K \times N_{\text{train}}$ is the total number of predictions in the dataset, $K = 128$ is the number of decoder queries per sample, and $N_{\text{train}} = 8389$ is the number of training samples. Unnormalized weights are given by $w'_c = 1/\sqrt{n_c}$ and are then normalized to sum to one via $w_c = w'_c / \sum_{c'=0}^{4} w'_{c'}$. This results in the final weight vector $[0.048,\ 0.403,\ 0.096,\ 0.231,\ 0.222]$, which is applied in the class-weighted cross-entropy loss.

**Chamfer loss computation.** Both $S_i^{c_i}$ (ground truth) and $S_j^{c_i}$ (prediction) contain 64 uniformly sampled points along a curve of class $c_i$, where sampling is performed on the respective curve type using the parameters of $i$ for $S_i^{c_i}$ and the parameters of $j$ for $S_j^{c_i}$. For example, if $c_i = 4$ (Bézier), we use the Bézier control points $\mathbf{B}_i$ and $\hat{\mathbf{B}}_j$ to sample the corresponding point sets. If $c_i = 2$ (line segment), we take the line parameters $\mathbf{L}_i$ and $\hat{\mathbf{L}}_j$ to form $S_i^1$ and $S_j^1$. If $c_i = 3$ (circle), the sets $S_i^3$ and $S_j^3$ are generated from the circle parameters $\mathbf{C}_i$ and $\hat{\mathbf{C}}_j$. Finally, if $c_i = 2$ (arc), we use the arc parameters $\mathbf{A}_i$ and $\hat{\mathbf{A}}_j$. In each case, sampling is performed uniformly along the specified curve type, ensuring that both sets contain exactly 64 points.

## C. Evaluation

We define the Chamfer distance between two 3D point sets $X$ and $Y$ as

$$\begin{aligned} \text{CD}(X, Y) = &\frac{1}{|X|} \sum_{\mathbf{x} \in X} \min_{\mathbf{y} \in Y} \|\mathbf{x} - \mathbf{y}\|_2^2 + \\ &\frac{1}{|Y|} \sum_{\mathbf{y} \in Y} \min_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{y}\|_2^2. \end{aligned} \tag{10}$$

We define the Hausdorff distance between two 3D point sets X and Y as

$$\text{HD}(X, Y) = \frac{1}{2} \Big( \max_{\mathbf{x} \in X} \min_{\mathbf{y} \in Y} \|\mathbf{x} - \mathbf{y}\|_2 + \max_{\mathbf{y} \in Y} \min_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{y}\|_2 \Big). \tag{11}$$

### C.1. Runtime

| N | Model ↓ | Post-processing ↓ |
|---|---|---|
| 32,768 | 0.1181 (± 0.02) | 0.0705 (± 0.10) |
| 16,384 | 0.0723 (± 0.02) | 0.0693 (± 0.10) |
| 8,192 | 0.0502 (± 0.02) | 0.0696 (± 0.10) |
| 4,096 | 0.0359 (± 0.02) | 0.0735 (± 0.10) |

Table 6. Inference and post-processing time measurement in seconds (NVIDIA RTX 4090) using different input point counts $N$ and $K = 256$.

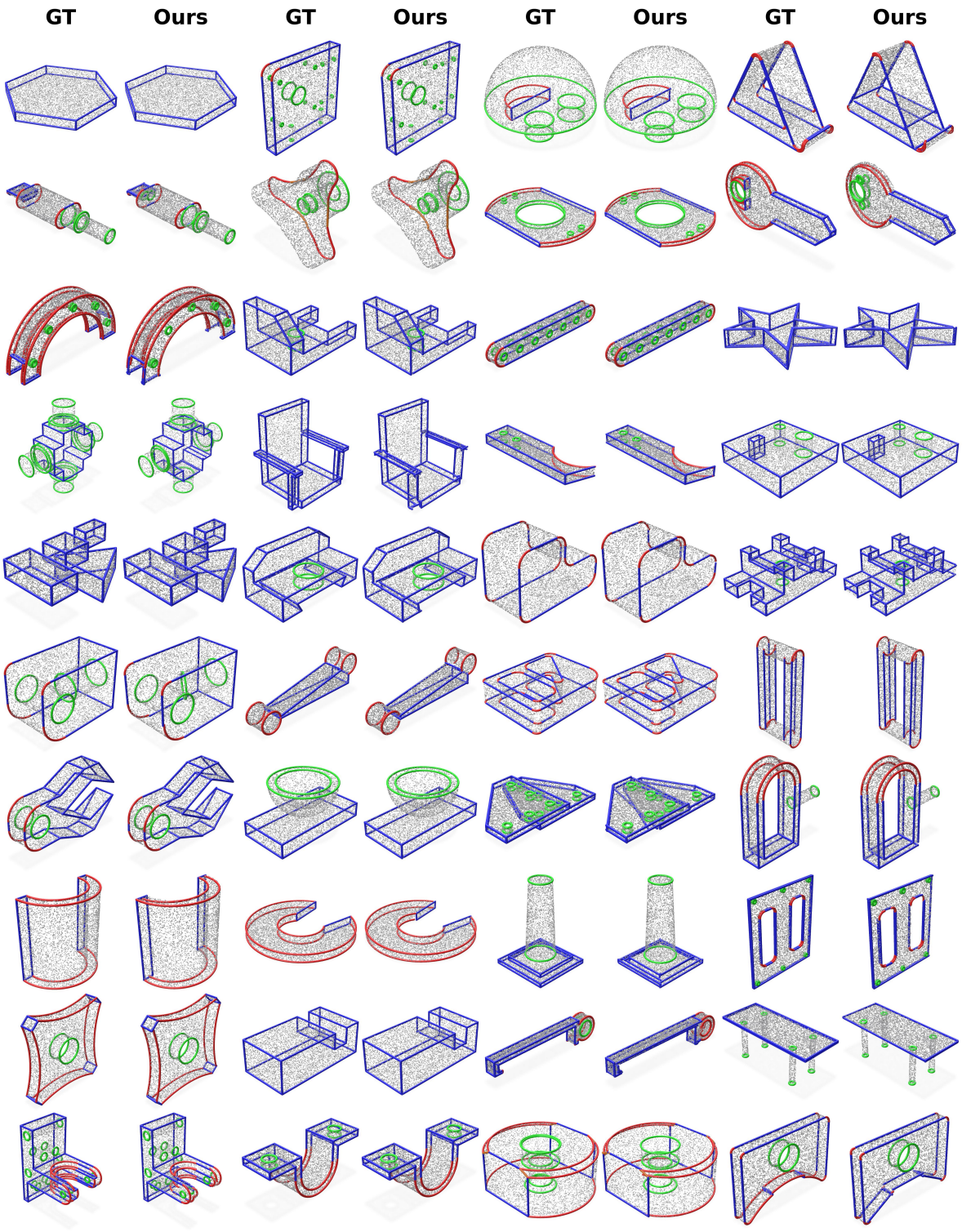


Figure 9. Qualitative visualization of PI3DETR's performance on various models from the ABC Dataset, shown in comparison to the corresponding ground truth. Each example is evaluated with a point sampling of $N = 32{,}768$ using best-performing trained model.