SymTr - Symmetry detection Transformer

Ido Raizman
Computer Science
Technion
ido.raizman@campus.technion.ac.il

Adam Gaber
Computer Science
Technion
adam.gaber@campus.technion.ac.il

Code available at: https://github.com/idoraizman/SymTr-Symmetry_Transformer

Abstract

The positional encoding mechanism in transformers is designed to break the order equivariance inherent in the attention mechanism, making it a fundamental component of transformer architectures. Point clouds, however, are unordered sets, which makes them a natural test case for evaluating transformer models without positional encoding. In this work, we adopt a DETR (Detection Transformer)-like encoder–decoder architecture, remove the positional encoding, and apply the model to the problem of detecting symmetry lines within point clouds with different backbones. Our findings suggest that, in this setting, transformers can learn effectively without explicit positional encodings. Performance improves further when the input consists of richer feature representations, such as the local and global descriptors provided by PointNet++.

1 Introduction

Symmetry is a fundamental cue in perception, with applications in recognition, parsing, and compact shape representation. While classical approaches relied on handcrafted descriptors or optimization in transformation space, recent advances in deep learning enable direct, data-driven symmetry detection.

Transformers, and DETR in particular, show that structured outputs can be learned as sets without handcrafted heuristics, while point-cloud models like PointNet and PointNet++ provide permutation-invariant embeddings. Because point clouds are unordered, they offer a natural setting to test transformers without positional encodings and to ask whether attention alone can capture geometric relations.

In this work we adapt a DETR-style architecture for detecting symmetry lines in 2D point clouds, intentionally omitting positional encodings. Our study highlights the conditions under which attention can encode symmetry cues and the importance of rich, locality-aware embeddings for effective learning, laying the groundwork for future extensions to 3D symmetry detection.

2 Related Work

Symmetry detection has progressed from exact, descriptor-driven pipelines to learning-based set prediction on point clouds. Classical methods targeted *perfect* symmetries; the seminal work of Mitra et al. (2006) reformulated detection as clustering in a transformation space via Hough-style voting and mean shift, enabling robust *partial/approximate* symmetries but assuming relatively clean data.

Generative-inspired approaches connect this view with score-based methods. Je et al. (2024) treat transformation space as a score field and use Riemannian Langevin dynamics for mode seeking, improving robustness to noise and incompleteness while handling reflections, rotations, and translations without supervision.

Transformers offer a complementary path for structured outputs. DETR (Carion et al., 2020) casts detection as set prediction with learned queries and Hungarian matching, naturally supporting variable-size outputs and geometric relations—an appealing fit when symmetries are treated as set elements.

For point-set representations, PointNet (Qi et al., 2017) learns from unordered points via shared MLPs and symmetric pooling, capturing global shape but missing fine local structure; PointNet++ (Qi et al., 2017) adds hierarchical neighborhoods to recover local–global context and has become a standard backbone. Point-cloud transformers further explore attention on unordered sets. PCT (Guo et al., 2021) embeds raw coordinates and relies on global pooling without explicit positional encodings. Studies of attention without positional encodings (Haviv et al., 2022; Zuo et al., 2025) show that relative structure can be learned from embeddings alone.

3 Methodology/architecture

3.1 Problem Setup

Given a 2D point set $X = \{x_i \in \mathbb{R}^2\}$, our goal is to predict up to P reflective symmetries as lines in normal form (n_x, n_y, d) with equation

$$n \cdot x + d = 0.$$

If $\mathbf{n}$ is normalized ($|\mathbf{n}| = 1$), it lies on the unit circle and can be parameterized as

$$\mathbf{n} = (\cos\theta, \sin\theta)$$

where θ is the angle the normal makes with the x-axis. Then d represents the signed perpendicular distance from the origin to the line along $\mathbf{n}$. Note that the line defined by $(\mathbf{n}, d)$ is identical to the line defined by $(-\mathbf{n}, -d)$, creating a sign ambiguity. To remove this, we conventionally restrict $\theta \in [0, \pi]$, so $\sin\theta \geq 0$, ensuring a unique representation.

3.2 Data Generation

We synthesize training data by:

1. Sampling K random symmetry line in normal form with unit normal $n = (\cos\theta, \sin\theta)$ and offset d.
2. Generating a random polynomial curve around each line and reflecting it across the symmetry line to create an ensample of symmetric shapes.
3. Extracting an outline via rasterization and contour detection, then sampling a fixed number of points along the polygonal outline.
4. filter out symmetry lines that are not well represented on resulting contour (less then $threshold$ fraction of the shape outline is part of the contour)

This results in a complex shape with diversely sized partial symmetries.

Shapes are normalized into $[-1, 1]^2$ and ground-truth lines are transformed accordingly. The dataset is saved as pairs $(points, lines)$ in `dataset.npz`.
to break the ambiguity between the symmetry line representation $(n, d), (-n, -d)$ we force the choice $\theta \in [0, \pi]$ meaning $n[1] = sin(\theta) > 0$

3.3 Backbone and Tokenization

We test 3 options for a backbone that creats embeddings for our model; raw coordinates, PointNet, PointNet++.

For the PointNet we use a 2D adaptation of the PointNet model and feed into it raw coordinates, the model is not pretrained and its parameters are learnt during training.

For the PointNet++, Each input point cloud $X \in \mathbb{R}^{B\times N\times 2}$ is lifted to a 9-channel tensor by concatenating zero-z and zero-color features for compatibility with a pretrained PointNet++ backbone.

The backbone outputs per-point embeddings that go through a linear layer to be projected into the Transformer hidden dimension D. We use P learned query embeddings as decoder tokens. analogous to DETR, these queries will undergo gradual refinement during cross attention with encoded pointcloud information and self attention among themselves as to finally become the decoder output from which line parameters are predicted.

3.4 Transformer Encoder–Decoder

On top of the backbone features, we build a Transformer encoder–decoder with L_e encoder layers and L_d decoder layers, n_{heads} attention heads, and hidden size D. Source tokens represent per-point features, while queries correspond to candidate symmetries. Decoder outputs $H \in \mathbb{R}^{B\times P\times D}$ are mapped via a two-layer MLP to $\mathbb{R}^{3+1}$ (line parameters and confidence).
Decoder output postprocessing: we clip the 2nd element to minimum 0 to enforce the decision made in the data generation to break line representation ambiguity.

$$n, d, conf = norm(out[0], max(out[1], 0)), out[2], out[3]$$

3.5 Set Prediction and Matching

Training follows DETR's set prediction principle. Predicted queries are matched to ground-truth lines using a geometry-aware cost:

$$cost = w_{dir}\,(1 - |\hat{n} \cdot n|)\ + |\hat{d} - d|,$$

The matching is solved with the Hungarian matching algorithm.

3.6 Loss Functions

Given the bipartite matches, we optimize two losses:

1. **Line regression:**
$$\mathcal{L}_{lines} = \lambda_{dir}\,(1 - \hat{n} \cdot n) + |\hat{d} - d|.$$
Normals are unit-normalized; We take the mean of all matched pair losses within the sample and return zero if no ground truth exists.
2. **Confidence classification:** A BCE-with-logits loss supervises confidence logits, assigning 1 for matched queries and 0 for unmatched queries.

3.7 Training Protocol

We implement the model in PyTorch Lightning with reproducible random seeds. Data is split 80/10/10. Optimization uses AdamW and a StepLR scheduler. For our hyperparameter tuning process, we train our models on a dataset of 10k samples and the final parameters are used to train the final model with a 20k sample dataset.

3.8 Outputs

At inference, the model returns P line hypotheses with confidences. Each line (n_x, n_y, d) is visualized by converting to endpoints along the direction perpendicular to n over the extent of the point set.

3.9 Summary

In summary, our architecture combines:

- A backbone for point embeddings, Raw/PointNet/Pointnet++
- A Transformer encoder–decoder for global reasoning,
- A DETR-style set prediction framework with Hungarian matching,
- Geometry-aware line regression and confidence losses.

4 Experiments

We evaluate our encoder–decoder transformer for symmetry detection on 2D point clouds.

1. Raw coordinates as tokens. Feeding raw $\mathbf{x}, \mathbf{y}$ coordinates as token embeddings allowed the transformer to converge to a valid solution, but performance was limited. Absolute coordinates provide only pointwise descriptions and lack explicit relational cues, so the model's ability to capture symmetry was constrained, leading to noisier and less accurate predictions.

2. PointNet embeddings. With features from a trainable PointNet backbone, convergence remained stable and results improved slightly over raw coordinates. PointNet enriches point features with some global context, but its heavy reliance on global aggregation diminishes explicit local geometric cues. Consequently, while the model learned valid symmetries, its accuracy was only moderately better.

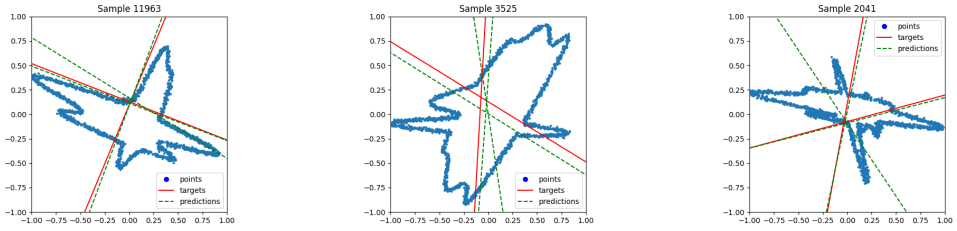
3. PointNet++ (hierarchical) embeddings. We used a pretrained model (taken from `https://github.com/yanx27/Pointnet_Pointnet2_pytorch/tree/master`). PointNet++ yielded a small but consistent improvement over raw coordinates and PointNet. Its hierarchical design provides both local and global geometric cues, which offered the transformer slightly better support for relational reasoning. Predictions were modestly more accurate and aligned somewhat better with the ground truth. As input shapes become more complex, we anticipate the backbone choice may matter more, with locality-aware embeddings being a reasonable direction to explore. In addition, PointNet++ produced more symmetry-representative attention maps, further suggesting its embeddings capture relational structure more effectively than the baselines.

5 Results

	Transformer Only	PointNet backbone	PointNet++ backbone	pointNet only	PointNet++ only
Offset error	0.065511552	0.064798884	0.056808617	0.086478194	0.090619846
Angle error	0.022869292	0.027847134	0.018474612	0.055621209	0.034833916

Table 1: Different Models results comparison

Table 1 errors computed after Hungarian matching between predicted and ground-truth lines. Each line is parameterized as $(\cos\phi, \sin\phi, d)$ with normal form $\mathbf{n}\cdot\mathbf{x} + d = 0$. We then compute: (i) *angle error* as the mean of $1 - \hat{\mathbf{n}}\cdot\mathbf{n}$ (i.e., $1 - \cos\theta$), and (ii) *offset error* as the mean absolute difference $|\hat{d} - d|$.



(a) Prediction 1 (b) Prediction 2 (c) Prediction 3

Figure 1: Example predictions using the PointNet++ backbone model.

(a) Attention maps for query 1

(b) Attention maps for query 2

Figure 2: Comparison of attention maps for different query predictions with the PointNet++ backbone.

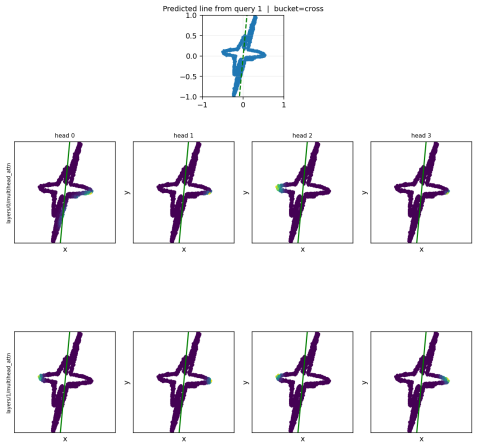


Figure 3: Attention map for a query prediction with the raw coordinate inputs, compared to figure 2(b) it's not as expressive in terms of the points which define the symmetry.

6 Summary

Analysis. Transformers without positional encodings rely entirely on their input embeddings to express pairwise relations via dot-product attention; there is no separate mechanism that injects order or relative geometry. Because symmetry detection is inherently relational (pairings, mirror correspondences, global axes), effective inference benefits from embeddings that expose such relations. In our experiments, all configurations converged to valid—though imperfect—solutions. Raw coordinates and PointNet features offered limited relational signal, whereas hierarchical, neighborhood-aware embeddings (PointNet++) provided a small but consistent gain by encoding more local and multi-scale geometric cues while also producing more symmetry-representative attention maps . We expect the backbone's impact to grow as the geometric structure becomes more complex.

Scope and limitations. All experiments were conducted on procedurally generated synthetic 2D point clouds with known ground-truth symmetries. This setup let us control shape composition, sampling density, and noise, and evaluate against exact targets. However, it lacks real-world artifacts.

Overall, our experiments on synthetic point clouds demonstrate that transformers without positional encodings can be effective for geometric reasoning, especially when combined with locality-aware backbones like PointNet++ that provide multi-scale structure. Still, results on synthetic data limit generality, and future work should validate on real scans and richer 3D geometries, explore better relational embeddings and attention mechanisms, and test larger models and auxiliary objectives to better capture relational structure.

References

[1] Mitra, N.J., Guibas, L.J. & Pauly, M. (2006) Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568. New York: ACM Press.

[2] Je, J., Liu, J., Yang, G., Deng, B., Cai, S., Wetzstein, G., Litany, O. & Guibas, L. (2024) Robust symmetry detection via Riemannian Langevin dynamics. In *Proceedings of SIGGRAPH Asia 2024 Conference Papers*, pp. 1–13. New York: ACM Press.

[3] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. & Zagoruyko, S. (2020) End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 213–229. Cham: Springer.

[4] Qi, C.R., Su, H., Mo, K. & Guibas, L.J. (2017) PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660. Los Alamitos, CA: IEEE Computer Society.

[5] Qi, C.R., Yi, L., Su, H. & Guibas, L.J. (2017) PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5099–5108. Red Hook, NY: Curran Associates, Inc.

[6] Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R.R. & Hu, S.-M. (2021) PCT: Point Cloud Transformer. In *Computational Visual Media*, Vol. 7, No. 2, pp. 187–199.

[7] Haviv, A., Ram, O., Press, O., Izsak, P. & Levy, O. (2022) Transformer Language Models without Positional Encodings Still Learn Positional Information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1382–1390. Abu Dhabi, UAE: Association for Computational Linguistics.

[8] Zuo, C., Guerzhoy, P. & Guerzhoy, M. (2025) Position Information Emerges in Causal Transformers Without Positional Encodings via Similarity of Nearby Embeddings. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, pp. 9418–9430. Abu Dhabi, UAE: Association for Computational Linguistics.