Eurographics Workshop on 3D Object Retrieval (2023)
S. Biasotti, M. Daoudi, U. Fugacci, G. Lavoué and R. C. Veltkamp (Editors)

Short Paper

SHREC 2023: Detection of symmetries on 3D point clouds representing simple shapes

Ivan Sipiran[1], Chiara Romanengo[2], Bianca Falcidieno[2], Silvia Biasotti[2], Gerasimos Arvanitis[3], Chen Chen[4,5], Vlassis Fotis[3], Jianfang He[4,5], Xiaoling Lv[4,5], Konstantinos Moustakas[3], Silong Peng[4,5], Ioannis Romanelis[3], Wenhao Sun[4,5], and Christoforos Vlachos[3], Ziyu Wu[4,5], Qiong Xie[5]

[1]Department of Computer Science, University of Chile, Santiago, Chile
[2]Istituto di Matematica Applicata e Tecnologie Informatiche 'E. Magenes' - CNR, Italy
[3]Department of Electrical and Computer Engineering, University of Patras, 26504 Patra, Greece
[4]School of Artificial Intelligence, University of Chinese Academy of Sciences, Huairou 101408, China
[5]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Abstract

This paper presents the methods that participated in the SHREC 2023 track focused on detecting symmetries on 3D point clouds representing simple shapes. By simple shapes, we mean surfaces generated by different types of closed plane curves used as the directrix of a cylinder or a cone. This track aims to determine the reflective planes for each point cloud. The methods are evaluated in their capability of detecting the right number of symmetries and correctly identifying the reflective planes. To this end, we generated a dataset that contains point clouds representing simple shapes perturbed with different kinds of artefacts (such as noise and undersampling) to provide a thorough evaluation of the robustness of the algorithms.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Shape analysis; Point-based models;***

1. Introduction

Symmetry aids in reducing complexity by representing an entity with less information. As a consequence, the analysis of symmetries is appealing in the search for more compact representations of objects in computer systems. However, computational models do not inherently possess information about their potential symmetries. Therefore, it is necessary to analyse these representations and detect symmetries starting from the available data. From a computational perspective, this symmetry detection task is challenging and has recently garnered attention from research communities in various areas, such as computer vision and geometry processing.

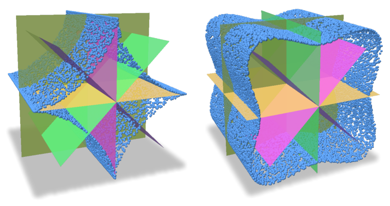


Figure 1: *Some shapes in our dataset and their symmetry planes.*

Although the study of symmetries has attracted significant interest from various research communities, particularly in the field of geometry processing, there remains a lack of consensus regarding the evaluation and comparison of symmetry detection algorithms. Each algorithm (such as those proposed in [SGS14; NR20; SHZ*20a; GZM*21; HKLM22]) employs a unique set of objects and metrics for evaluation, making it nearly impossible to compare the advantages and limitations of the methods. The present paper aims to address this issue by introducing a large-scale dataset for the assessment of symmetry detection algorithms in 3D data. Our approach involves the synthetic generation of 3D shapes, whose reflective symmetries are derived analytically. As a result, the proposed dataset offers high-quality ground truth (see Fig. 1), which can be leveraged by recent methods, including those based on machine learning.

There are few datasets concerning symmetries. For instance, for images, we mention Tsogkas et al. [TK12] and Seo et al. [SKKC22]. The latter proposed a dataset for reflection and rotation symmetry detection in real-world images, called DENse and DIverse symmetry (DENDI), with accurate and clean annotations for reflection and rotation symmetries. Regarding symmetries in three-dimensional shapes, [SWZ*18] introduces a large-scale dataset of diverse image-shape pairs called Pix3D that associates a rich set of images to each 3D shape, equipped with an accurate 3D pose annotation to ensure precise 2D-3D alignment. Finally, ShapeNet [CFG*15] contains 3D CAD models annotated with geometric at-

DOI: 10.2312/3dor.20231148

tributes such as rigid alignments, parts and key points, symmetries (reflection plane, other rotational symmetries), and physical sizes.

None of the datasets developed for symmetry detection focused specifically on point clouds. Recently, the SHREC'22 dataset of 3D segments [RRB*22] introduced parts of primitives represented as point clouds. Different from our goal, that benchmark was devoted to primitive and parameter identification, rather than symmetry detection. In contrast, recent machine learning-based methods (such as those proposed in [SHZ*20a; GZM*21]) typically employ well-known 3D datasets like ShapeNet [CFG*15] or ModelNet [WSK*15] and derive ground truth from prior knowledge within the dataset or using semi-automatic annotation tools.

This SHREC track aims to assess the effectiveness of automatic algorithms in detecting symmetries in 3D point clouds that represent simple shapes. The objective is to determine the reflective planes for each point cloud. By simple shapes, we refer to surfaces generated by various types of closed, plane curves that are projected in multiple ways. Specifically, we generate cylinders or cones having different plane curves as directrix. We also apply different kinds of perturbations (such as noise and undersampling) to the point clouds to evaluate the robustness of methods to detect symmetries under these transformations.

For each point cloud of the test set, the participants had to provide the number of the detected symmetries, the normal vectors of the detected planes, and a point belonging to each reflective plane. We also asked participants to provide a confidence value for each detected plane, which is used in the evaluation metrics. The performance of the algorithms is evaluated based on the quality of the recognised planes and the number of planes correctly identified. This analysis is specified according to the type of point cloud artefacts to provide a thorough evaluation of the robustness of the algorithms.

The rest of the paper is organised as follows. Section 2 is focused on the description of the proposed benchmark, specifically the dataset, the ground truth, and the evaluation measures necessary to quantify the methods' performances. Section 3 describes the three automatic algorithms that participate in this SHREC track. Section 4 lists the setups and conducted experiments. Finally, Section 5 provides the comparative analysis of the results obtained by the methods.

2. The benchmark

In this section, we describe the curves used to generate our dataset, and the resulting dataset (Section 2.1), the ground truth files associated with the point clouds of the training set (Section 2.2) and the evaluation metrics we adopted (Section 2.3).

2.1. Dataset

We have selected some families of closed plane curves used as directrix of the cylinders or cones in our dataset. Specifically, we have chosen them within an atlas of curves [Shi95] since they are characterised by distinctive shapes. Table 1 depicts the family of 2D curves used to create our dataset and their parametric representation. Once a curve of a chosen family is selected, it is possible to generate a cylinder or a cone with this curve as a directrix. Our procedure extrudes the curve lying in the XY plane along the Z axis.

The dataset is composed of $69,000$ three-dimensional simple shapes represented as point clouds; it is divided into a training set and a test set that contain, respectively, $60,000$ and $9,000$ point clouds.

The point clouds considered in this benchmark were built by means of the following procedure. First, we randomly choose one of the closed plane curves described in Table 1 and randomly select the parameters for the generation. Specifically, we set the parameters of each curve as follows:

- Citrus: $a = 1$ and b randomly chosen in the interval $[1, 13]$, $b \in \mathbb{N}$.
- m-Convexities: a randomly chosen in the interval $[0.5, 1.1]$, $a \in \mathbb{R}$, b randomly chosen in the interval $[0.2, 0.9]$, $b \in \mathbb{R}$, and m randomly chosen in the interval $[3, 9]$, $m \in \mathbb{N}$.
- Geometric petal: a randomly chosen in the interval $[1.0, 2.0]$, $a \in \mathbb{R}$, b randomly chosen in the interval $[1, 6]$, $b \in \mathbb{N}$, and m randomly chosen in the interval $[1, 6]$, $m \in \mathbb{N}$.
- Lemniscate of Bernoulli: $a = 1$.
- Egg of Keplero: $a = 1$.
- Mouth curve: $a = 1$.
- Astroid: $a = 1$.

Then, we generate the point cloud through the extrusion procedure, selecting between a cylindrical or conical extrusion. Since both kinds of shapes have the rotational axis coincident with the z-axis, we randomly apply translations and/or rotations to the point cloud to make its position and orientation more generic. Finally, each point cloud is transformed with the following perturbations:

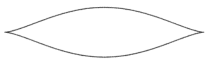
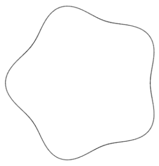
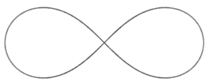
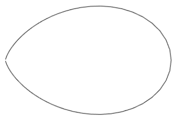
- *P0-Clean.* The point cloud is not perturbed.
- *P1-Uniform noise of different intensities.* A random percentage, between 30% and 80%, of the point cloud, is perturbed, by applying a uniform noise, obtained by sampling uniform distributions of the form $\mathcal{U}(-\frac{1}{n}, \frac{1}{n})$, being n a random value among $15, 17, 19, 20$.
- *P2-Gaussian noise of different intensities.* A random percentage, between 30% and 80%, of the point cloud, is perturbed by applying a Gaussian noise, obtained by sampling Gaussian distributions of the form $\mathcal{N}(0, \frac{1}{n^2})$, being n a random value among $20, 23, 27, 30$.
- *P3-Undersampling.* A random percentage, between 30% and 80%, of the point cloud is removed.
- *P4-Combination of uniform noise of different intensities and undersampling.*
- *P5-Combination of Gaussian noise of different intensities and undersampling.*

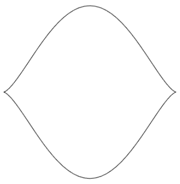
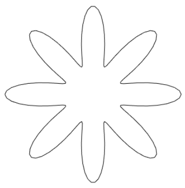
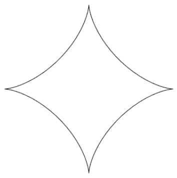
For the noise transformations, we first decide how many points will be perturbed with a probability between 30% and 80%. For each point to perturb, we generate a random shift in the distribution (uniform or Gaussian) with the value of n also chosen randomly.

2.2. Ground truth

For each point cloud of the training set, we provide the ground-truth information for the symmetric planes. Each plane is represented by a normal vector and a point belonging to the symmetric plane. In

Table 1: *Families of plane curves used as directrix of the cylinder or cone in our dataset.*

citrus curve	$m-$convexities	lemniscate of Bernoulli	egg of Keplero
 $\mathbf{P}(t) := \begin{cases} x = t - \frac{a}{2} \\ y = \pm\sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases}$	 $\mathbf{P}(t) := \begin{cases} x = \frac{a}{1+b\cos(mt)}\cos t \\ y = \frac{a}{1+b\cos(mt)}\sin t \end{cases}$	 $\mathbf{P}(t) := \begin{cases} x = a\frac{\sin t}{1+\cos^2 t} \\ y = a\frac{\sin t\cos t}{1+\cos^2 t} \end{cases}$	 $\mathbf{P}(t) := \begin{cases} x = \frac{a}{(1+t^2)^2} \\ y = \frac{at}{(1+t^2)^2} \end{cases}$

mouth curve	geometric petal	astroid
 $\mathbf{P}(t) : \begin{cases} x = a\cos t \\ y = a\sin^3 t \end{cases}$	 $\mathbf{P}(t) := \begin{cases} x = (a+b\cos nt)\cos t \\ y = (a+b\cos nt)\sin t \end{cases}$	 $\mathbf{P}(t) := \begin{cases} x = a\cos^3 t \\ y = a\sin^3 t \end{cases}$

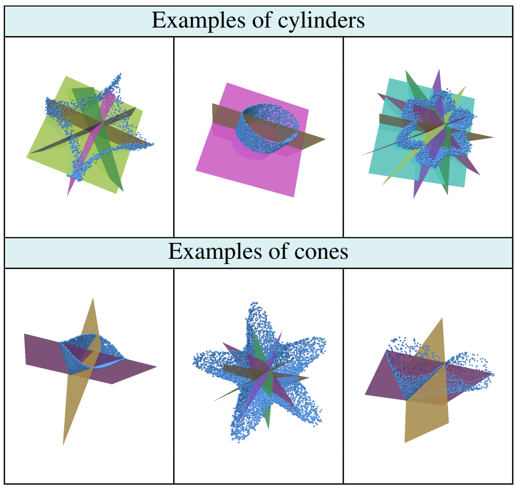


Figure 2: *A visual illustration of point clouds in our training set with their planes of symmetry.*

Figure 2 we show some point clouds of the training set together with their planes of symmetry. For each point cloud of the test set, the track participants were required to return a file with the predicted planes. In addition, for each detected symmetry plane, we ask the participants to provide a confidence of the detected symmetry in the range $[0,1]$. The confidence is used to rank the detections and compute the evaluation metrics. If a method does not compute it, a value of 1.0 is entered.

2.3. Evaluation measures

To assess the effectiveness of automatic algorithms in detecting symmetries on point clouds in varying conditions, we focus on the quality of the recognised planes and the accuracy of the identified planes. To comprehensively evaluate the robustness of these methods, this analysis is specific according to the type of artifact considered.

To compare the algorithms, two metrics are employed: mean average precision (mAP) and precision at highest confidence (PHC). These metrics share similarities with the evaluation of object detection methods in computer vision [LOW*20]. Both metrics are calculated based on a confidence value for the detections, which facilitates the identification of reliable plane detections.

- The *mean Average Precision* (mAP) is a metric used to evaluate the performance of a complete set of symmetries. Given a point cloud $\mathcal{P}$ and a set of ground-truth symmetry planes $S^{\mathcal{P}}$, as well as a set of symmetry planes $D^{\mathcal{P}}$ detected by an algorithm, the set $D^{\mathcal{P}}$ is sorted according to the confidences, and a correspondence is established between $S^{\mathcal{P}}$ and $D^{\mathcal{P}}$ based on the proximity of their respective planes in terms of angle and position.
  More specifically, a plane $s \in S^{\mathcal{P}}$ is deemed to match a plane $d \in D^{\mathcal{P}}$ if

  $$angle(\vec{N}_s, \vec{N}_d) \leq \theta \text{ and } distPlane(point(d), s) \leq \varepsilon$$

  where $angle(\vec{N}_s, \vec{N}_d)$ represents the angle between the normal vector to the plane s and the normal vector to the plane d, and $distPlane(point(d), s)$ represents the point to plane distance between the predicted point and the ground truth plane.

In this context, a match is considered a true positive, while detected planes without matches are false positives. This information is then used to calculate the average precision, and subsequently, the mean average precision is computed as the mean value across the entire test set.
The *Precision at Highest Confidence* (PHC) metric is defined as the average number of test shapes where the detection algorithm identifies a valid symmetry with the highest confidence. To illustrate the idea of this metric, let's suppose we have a test shape T with its detected symmetry planes S_T. We take the detected symmetry plane with the highest confidence and check if this symmetry matches with any symmetry in the ground-truth. If a match occurs, assign a value of one for the test shape and a value of zero otherwise. The final result for the metric is the average of assigned values for the entire test set. Therefore, this metric evaluates the effectiveness of an algorithm in detecting symmetries with the utmost level of confidence.

3. Description of the methods

3.1. M1: Learning to Predict Reflectional Symmetries with Point Cloud Transformer

Method M1 follows the dense prediction scheme of SymmetryNet [SHZ*20b]. It considers Point Cloud Transformer (PCT) [GCL*21] as its backbone and keeps three heads for predicting M reflectional symmetry normal vectors, one object centre, and M reflectional symmetry confidences. Figure 3 shows the pipeline of this method. It consists of three main steps:

1. *Farthest point sampling.* The input point cloud is preprocessed using the farthest point sampling method (see [QYSG17]) to obtain a uniform sampling of it. The point cloud is normalised by centring it to the origin and scaling it into a unit sphere.
2. *Dense symmetry parameter prediction.* Similarly to SymmetryNet, the proposed approach extracts point-wise geometric features from an input point cloud using PCT. Then the point-wise geometric features are fed to a spatially weighted pooling layer to extract a global feature that is then concatenated with the point-wise features for point-wise prediction tasks. Finally, after making predictions for each individual point of the point cloud, all the predictions are aggregated to form the final one. The total loss for an input point cloud is $\mathcal{L} = \frac{1}{N}\sum_i^N \mathcal{L}_i$, where
$$\mathcal{L}_i = \mathcal{L}_i^{conf} + \mathcal{L}_i^{vec} + \mathcal{L}_i^{cent} \tag{1}$$
is the prediction loss for the $i-$th point and N. is the number of points of the point cloud. For all the M predicted reflectional symmetry normal vectors, the approach first matches them to K, with $K \leq M$, ground-truth reflectional symmetry normal vectors. Following SymmetryNet, the positions of the K matched predicted normal vectors are recorded to form a K-hot ground truth label vector T_i for the predicted confidence vector C_i with length M. Then the confidence loss is computed as:
$$\mathcal{L}_i^{conf} = \mathcal{L}^{cls}(C_i, T_i) \tag{2}$$
where $\mathcal{L}^{cls}$ is the binary cross entropy loss. For the K matched predicted normal vectors n_i^k, with $k = 1, \ldots, K$, this method measures the differences between them and the ground-truth normal vectors $\hat{n}_i^k$ in a max-margin manner:
$$\mathcal{L}_i^{vec} = \frac{1}{K}\sum_{k=1}^{K} max(0, 1 - |n_i^k \cdot \hat{n}_i^k|^2) \tag{3}$$
where $n_i^k \cdot \hat{n}_i^k = \{1, -1\}$ guarantees that the predicted normal vectors will be co-linear with the matched ground truths. Same to SymmetryNet, the method explicitly predicts the centre c_i of the point cloud as the point lying on the reflection symmetry plane, measuring the distance between this prediction and the ground-truth centre $\hat{c}$:
$$\mathcal{L}_i^{cent} = d^2(c_i, \hat{c}) \tag{4}$$
where d is the Euclidean distance.
3. *Post-processing.* All the NM predicted normal vectors are clustered into M vectors using the density-based spatial clustering (DBSCAN) [EKSX96]. The clustering is performed among N points. After filtering out predicted normal vectors with low confidence, non-maximum suppression (NMS) is implemented to eliminate predictions that are close to the more confident ones. The centring and scaling of pre-processing are inversely applied to the resulting prediction to form the final prediction.

The original training set is split into 5 folds to perform cross-validation. The maximum number of symmetry planes is searched over the original training set and set as M. Fig. 3 provides a graphical abstract of method M1.

3.1.1. Computational aspects

The method is trained and tested on a 64-bit Ubuntu OS system with an Intel(R) Xeon(R) Silver 4208 processor (at 2.1 GHz), 126 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti GPU with 11 GB of memory. The training is performed over about 77 epochs, with a training time of approximately 157 hours in total (2 hours per epoch). When inferring, the neural network inference time per cloud is 8.742 ms, and the post-processing time per cloud is 110.123 ms.

3.2. M2: Multi-Head Symmetry Transformer (MHST)

Method M2 uses a deep neural network to encode the input point cloud and then create several smaller heads, each tasked to map the feature vector to a symmetry plane. Since the number of symmetry planes is not a priori known, a sufficiently large number of heads is chosen to cover the edge cases. For simpler shapes that exhibit fewer symmetries, most heads converge to the same plane parameters, and some produce invalid planes. A cleaning algorithm discards duplicates as well as planes with high symmetry errors and identifies the correct number of planes for each specific shape. Figure 4 shows the pipeline of this method.

This method considers the plain vision Transformer as a backbone network and pretrains it on the ShapeNet [CFG*15] dataset for reconstruction, following the Masked Autoencoding pipeline, as described in [PWT*22]. Specifically, 64 centroid points are sampled, and a neighbourhood of 32 points is built for each one. A PointNet-like network is applied per neighbourhood to extract feature embeddings. To extract positional embeddings, the xyz coordinates of the centroids are passed through an MLP. By adding the

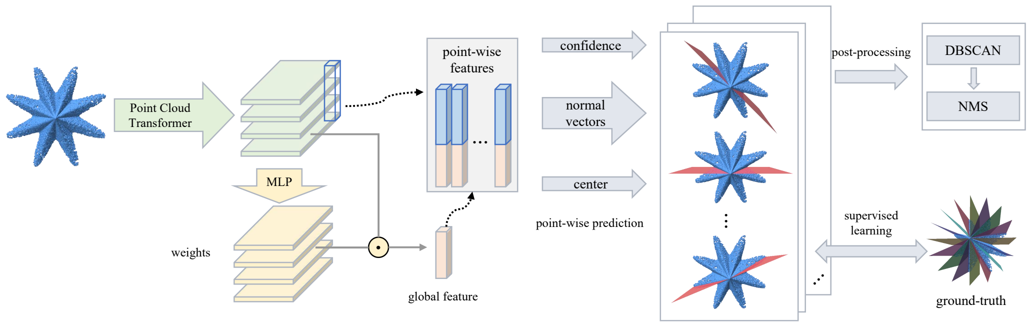


Figure 3: *A visual scheme of the strategy adopted in M1.*

positional and feature embeddings, the final patch embeddings are extracted and fed into the transformer. The Transformer is 12 transformer blocks deep, with 6 attention heads in each block, and the feature dimension equals to 384.

The global feature vector generated by the backbone is then sequentially fed into the network heads, which map the feature vector to a 4−dimensional vector of plane parameters. In the experiments, the best number of heads was 27. As a result, the network can generate up to 27 planes, assuming that no point cloud contains more symmetries. Each head consists of 3 linear layers with ReLU activations, batch normalisation, and dropout.

Since the number of output planes varies from shape to shape, this method follows an unsupervised approach inspired by Gao et al. [GZM*20]. The loss function is computed as

$$L = L_s + w_r L_r$$

where L_s is the symmetry term, L_r is the regularisation loss L_r that discourages the heads from converging to the same plane, and w_r is a weight fixed to 17. More in detail:

- the symmetry term L_s is computed as
$$L_s = \sum_{\pi_i} D(\mathcal{P}, \mathcal{P}'_i)$$
where $\mathcal{P}$ is the input point cloud, $\mathcal{P}'_i$ is a mirrored version of $\mathcal{P}$ whit respect to each predicted plane π_i and D is the Chamfer distance;
- the regularisation term is defined as:
$$L_r = \|MM^T - I\|_F^2$$
where $M = \begin{pmatrix} n_1 & n_2 & \cdots & n_{27} \end{pmatrix}^T$, n_i are the (normalised) normal vectors of each predicted symmetry plane π_i, I is the identity matrix (of size 27), and $\|\cdot\|_F$ is the Frobenius norm.

Finally, a cleaning algorithm discards duplicates to avoid trivial solutions. Specifically, all planes whose normal's angles were within $\frac{\pi}{10}$ radians were considered to be the same plane and only the one with the smallest symmetry loss was kept. Additionally, all planes whose symmetry loss was both (i) not within 1.35 times the average loss of the remaining planes, and (ii) larger than a threshold value (7 worked well for this particular configuration) were also removed. Fig. 4 provides a graphical abstract of method M2.

3.2.1. Computational aspects

The model was trained on a single RTX2060-6GB GPU. The CPU is an 8-core intel i7-9700k with a clock frequency of 3.6 GHz. The training time required was 3 hours. For inference, 60ms are required for processing a batch containing a single sample.

3.3. M3:Diffusion-based symmetry detection

This method is adapted from the approximated symmetry detection approach proposed in [SGS14]. Figure 5 shows the pipeline of this method. The method consists of three steps:

1. *Point cloud pre-processing.*
To begin, the method rescales the point cloud such that the diagonal of the object measures one. Next, it calculates the Laplacian of the point cloud using the robust Laplacian method developed by [SC20]. This technique produces a highly non-manifold triangulation from the input point cloud, which may result in a geometry with numerous degenerate triangles. To address this issue, the method introduces a small distortion in every triangle of the triangulation. This distortion is referred to as the mollification factor and can be set to higher values for greater robustness. In this case, the mollification factor is fixed at $1e-3$.
Finally, the algorithm performs the eigen-decomposition of the Laplacian to obtain ten eigenvalues along with their corresponding eigenvectors. As a result, the input point cloud is represented in the spectral domain by this set of eigenvalues λ_i and eigenvectors ϕ_i.
2. *Computation of symmetry planes.* Sipiran et al.[SGS14] propose a diffusion-based function on a surface that retains both local and global information while also revealing symmetric structures. The function is defined as follows
$$S(x,t) = \int_0^t h(x,t) = \sum_{i=0}^{\infty} \left(\frac{1 - e^{-\lambda_i t}}{\lambda_i} \right) \phi_i(x)^2 \qquad (5)$$
where $h(\cdot,\cdot)$ is the heat kernel restricted to the temporal domain. The points with high chances of having symmetric correspondences tend to be located in the local maxima of the function $S(x,t)$. Therefore, points in local maxima are excellent candidates for searching for symmetric planes. In the case of point clouds, a point v is considered a local maximum with respect to S if $S(v,t)$ is greater than $S(w,t)$ for every point w in the

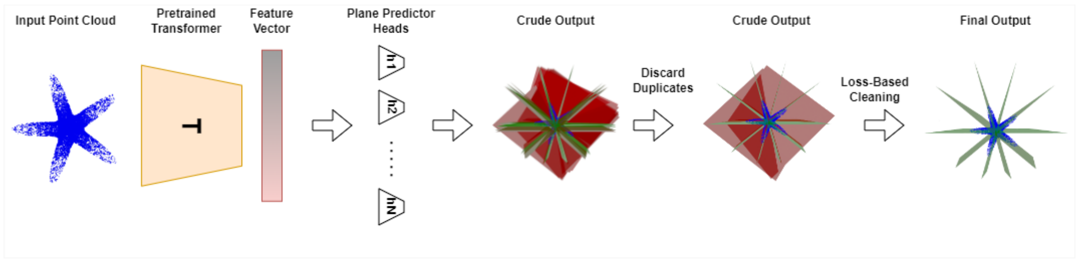


Figure 4: *A graphical representation of the strategy adopted in method M2.*

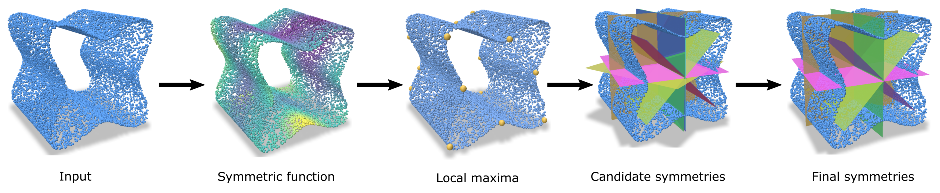


Figure 5: *A graphical representation of the strategy adopted in method M3.*

K-nearest neighbourhood. The set of selected points with local maximum values of S forms a potential candidate set.
Since it is not known which points are symmetric with others, the method generates a candidate symmetric plane between every pair of candidate points. Let v and m be two candidate points, the candidate symmetric plane is defined as $(\vec{n}, p)$ where $\vec{n} = (v-m)/|v-m|$ and $p = (v+m)/2$.

3. *Refinement.* To eliminate false positives and overlapping symmetric planes, the method implements a two-step filtering strategy. Firstly, the algorithm applies the symmetric transformation of each candidate plane to the input point cloud and computes the bidirectional Chamfer distance [ADMG17] between the original point cloud and the transformed point cloud. Only those candidate planes with a bidirectional Chamfer distance of less than 100 are selected. This filtering step helps to ensure that the candidate planes are real symmetric planes.
Secondly, the procedure filters out overlapping symmetric planes by sorting them according to their bidirectional Chamfer distance and performing a greedy selection. The symmetric plane with the lowest Chamfer distance is directly added to the final set. If the normal of a new symmetric plane deviates by more than five degrees from any of the symmetric planes in the final set, it is included in the final set; otherwise, it is discarded. The confidence value of a symmetric plane is computed as $1 - cd/100$, where cd is the bidirectional Chamfer distance. By assigning higher confidence values to the symmetric planes with lower Chamfer distances, we can prioritise the more accurate symmetric planes in the final result.

3.3.1. Computational aspects

The method was executed in a computer with a processor AMD EPYC 7282 16-Core, with 128GB RAM, under a Linux Ubuntu operating system. The processing of an input point cloud takes between 20 seconds and 1 minute; most of the time is spent in the eigendecomposition of the Laplacian matrix.

Method	mAP	PHC
M1 (R1)	0.0229	0.0225
M1 (R2)	0.3933	0.3448
M1 (R3)	0.9052	0.8950
M1 (R4)	**0.9707**	**0.9685**
M2	0.5403	0.2315
M3 (R1)	0.5251	0.5168
M3 (R2)	0.5160	0.5081

Table 2: *Overall performance of methods when the tolerance for angle deviation is $\theta = 1$ and distance-to-plane deviation is $\varepsilon = 1\%$ of the point cloud diagonal.*

4. Experimental Setup

We present the experiments performed with the three methods described in Section 3. In total, seven runs were executed. Here we describe the details of these experiments:

- M1 (R1). SymmetryNet [SHZ*20b] with reflectional symmetry loss and PointNet as backbone, trained for ten epochs with 5-fold cross validation.
- M1 (R2). SymmetryNet using the reflectional symmetry loss and an additional vector loss defined in Eq. 3 to directly fit the ground truth normal. The backbone is a Point Cloud Transformer (PCT), trained by one epoch with 5-fold cross validation.
- M1 (R3). SymmetryNet with loss defined in Eq. 1, PCT as backbone and trained by three epochs with 5-fold cross validation.
- M1 (R4). The same as M1 (Run 3), but trained by 77 epochs with 5-fold cross validation.
- M2. As described in Section 3.
- M3 (R1). The number of neighbors to detect local maxima is set to $K = 200$.
- M3 (R2). The number of neighbors to detect local maxima is set to $K = 250$.

Learning-based methods (M1 and M2) were trained on the train-

	Method	P0	P1	P2	P3	P4	P5
mAP	M1 (R1)	0.022	0.021	0.024	0.024	0.019	0.025
	M1 (R2)	0.392	0.416	0.392	0.388	0.402	0.367
	M1 (R3)	0.901	0.905	0.912	0.899	0.911	0.901
	M1 (R4)	**0.971**	**0.971**	**0.977**	**0.967**	**0.962**	**0.973**
	M2	0.548	0.536	0.535	0.551	0.529	0.541
	M3 (R1)	0.704	0.444	0.541	0.603	0.364	0.493
	M3 (R2)	0.696	0.432	0.537	0.590	0.354	0.485
PHC	M1 (R1)	0.022	0.021	0.024	0.024	0.018	0.025
	M1 (R2)	0.348	0.360	0.348	0.343	0.347	0.322
	M1 (R3)	0.893	0.892	0.905	0.890	0.901	0.887
	M1 (R4)	**0.972**	**0.968**	**0.977**	**0.964**	**0.956**	**0.972**
	M2	0.236	0.228	0.222	0.238	0.229	0.234
	M3 (R1)	0.704	0.435	0.534	0.599	0.347	0.480
	M3 (R2)	0.695	0.424	0.532	0.586	0.338	0.472

Table 3: *Performance of methods against transformations when the tolerance for angle deviation is* $\theta = 1$ *and distance-to-plane deviation is* $\varepsilon = 1\%$ *of the diagonal of the point cloud. P0, P1, P2, P3 and P5 denote the type of perturbation following the notation introduced in Section 2.1*

ing dataset and provided the detections on the test set for evaluation. As method M3 does not require a learning step, it was directly executed on the test set for the experiments.

5. Comparative analysis

This study presents the performance evaluation of the methods for identifying true positive detections using different thresholds and, possibly, multiple configurations. Our initial analysis considered a threshold of angular deviation at $\theta = 1$ and a threshold of distance-to-plane deviation at $\varepsilon = 1\%$ of the diagonal of the point cloud. The overall performance of the participant methods is displayed in Table 2. Furthermore, we evaluated the performance of the methods for the transformation applied to the point clouds, which can be seen in Table 3, and the kind of shape, which is illustrated in Table 4.

In this experiment, method M1 (run 3 and run 4) displays the highest mean average precision (mAP) and precision at highest confidence (PHC) metrics. This learning-based approach seems to exploit the scale of the dataset, the choice of architecture, and loss functions to learn the symmetric structure of the objects. In contrast, method M2 exhibits a decrease in precision, particularly a drop in the PHC metric, indicating that it is not accurate enough to assign the highest confidence to a correct plane. Method M3 shows the lowest mAP values among the competition, although close to the results of method M2.

Regarding transformations, learning-based methods (M1 and M2) have a uniform behaviour, indicating no significant differences across different transformations. This could be attributed to the high capacity of the trained models to learn some invariance of the symmetric structure of the point clouds concerning the transformations. Moreover, we observe that the dataset is built with the care of balancing the transformations. On the other hand, method M3 consistently performs worse in detecting planes under uniform and undersampling+uniform transformations. Since it is a geometric algorithm based on the heat diffusion process on the surface of the underlying object, the experiments show that the noise from the uniform distribution affects the computation of the symmetry planes.

The scenario is different when analysing the performance for the kind of shape. For example, method M1 struggles to detect symmetric planes in *m*-convexities shapes. We hypothesise that this behaviour can be attributed to the high variability in the number of symmetric planes in *m*-convexities shapes. This phenomenon is also slightly evident in the geometric petal shape, although the number of symmetries is not as variable as in *m*-convexities. Consistent with our findings, method M2 displays low mAP values in the geometric petal shape, unlike in the other types of shapes. Method M3 has a good mAP for the *astroid* shape but lower values for the rest of the shapes. Notably, it exhibits low precision in detecting planes in the *egg of Keplero* shape. The difference between these two different behaviours can be attributed to the nature of the algorithm. Method M3 detects symmetric key points that guide the entire process of detecting symmetric planes. The *astroid* shape contains well-defined key features that can be detected by the local maxima of diffusion functions. In contrast, the *egg of Keplero* shape does not have outstanding features, and method M3 struggles to find good symmetric candidates.

6. Concluding Remarks

This paper introduces a novel dataset and methodology for assessing the performance of symmetry-detection algorithms. The dataset consists of synthetic shapes represented as point clouds, giving the benefit of generating many shapes and creating a suitable scenario for learning-based methods. Moreover, the synthetic dataset facilitates the computation of a high-quality ground truth consisting of precisely computed symmetric planes.

This study evaluates three methods, with seven runs in total, two of which (M1 and M2) are based on neural networks for learning the symmetry planes, and one is a purely geometric method (M3). Our experimental results indicate a clear predominance of learning-based methods in identifying the orientation and location of symmetry planes, even in the presence of transformations that perturbed the input shape. We attribute this success to the combination of powerful learning architectures, such as Transformers, and a rigorous training protocol that takes advantage of a large number of shapes and their variability.

We identify several potential avenues for future research. First, the benchmark and the evaluation would benefit from including more challenging transformations, such as stronger noise and shape partiality. Second, since the dataset is generated synthetically, creating a larger dataset is possible. A large-scale dataset, one or two orders of magnitude larger than the current one, could push the limits of how algorithms operate and learn from 3D data. We anticipate that powerful neural networks are capable of handling such a magnitude of data, and a larger dataset may lead to the establishment of new challenges, not just for symmetry detection.

	Method	S1	S2	S3	S4	S5	S6	S7
mAP	M1 (R1)	0.0387	0.0161	0.0229	0.0097	0.0039	0.0698	0.0000
	M1 (R2)	0.4637	0.3290	0.2170	0.4037	0.7546	0.1431	0.4268
	M1 (R3)	0.9554	0.9778	0.6857	0.9254	0.9864	0.8031	0.9860
	M1 (R4)	**0.9766**	**0.9968**	**0.8542**	**0.9669**	**0.9996**	**0.9961**	**0.9976**
	M2	0.5129	0.5208	0.5139	0.4720	0.5396	0.7310	0.4954
	M3 (R1)	0.9009	0.3729	0.5104	0.4037	0.6354	0.2420	0.6061
	M3 (R2)	0.9009	0.3656	0.4858	0.3821	0.6354	0.2373	0.6002
PHC	M1 (R1)	0.0387	0.0146	0.0224	0.0089	0.0039	0.0698	0.0000
	M1 (R2)	0.3783	0.2919	0.1588	0.3222	0.7355	0.1325	0.3817
	M1 (R3)	0.9477	0.9877	0.6534	0.9308	0.9977	0.7286	0.9992
	M1 (R4)	**0.9803**	**1.0000**	**0.8462**	**0.9531**	**1.0000**	**0.9929**	**1.0000**
	M2	0.1137	0.2127	0.1804	0.1935	0.2087	0.5812	0.1341
	M3 (R1)	0.8908	0.3656	0.5029	0.3988	0.6168	0.2416	0.5970
	M3 (R2)	0.8908	0.3587	0.4788	0.3780	0.6168	0.2369	0.5915

Table 4: *Performance of methods with respect to the type of shape when the tolerance for angle deviation is $\theta = 1$ and distance-to-plane deviation is $\varepsilon = 1\%$ of the diagonal of the point cloud. S1=Astroid, S2=Citrus, S3=M convexities, S4=Geometric petal, S5=Lemniscate Bernoulli, S6=Egg Keplero, ans S7=Mouth curve.*

Acknowledgements

The work of Ivan Sipiran has been partially supprted by ANID Chile - Research Initiation Program - Grant N° 11220211.

References

[ADMG17] ACHLIOPTAS, PANOS, DIAMANTI, OLGA, MITLIAGKAS, IOANNIS, and GUIBAS, LEONIDAS J. "Learning Representations and Generative Models for 3D Point Clouds". *Int. Conf. on Machine Learning*. 2017 6.

[CFG*15] CHANG, ANGEL X., FUNKHOUSER, THOMAS, GUIBAS, LEONIDAS, et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015 1, 2, 4.

[EKSX96] ESTER, MARTIN, KRIEGEL, HANS-PETER, SANDER, JÖRG, and XU, XIAOWEI. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231 4.

[GCL*21] GUO, MENG-HAO, CAI, JUN-XIONG, LIU, ZHENG-NING, et al. "PCT: Point cloud transformer". *Computational Visual Media* 7 (2021), 187–199 4.

[GZM*20] GAO, L., ZHANG, L. -X., MENG, H. -Y., et al. "PRS-Net: Planar Reflective Symmetry Detection Net for 3D Models". *IEEE Trans. on Vis. and Computer Graph.* (2020), 1–1 5.

[GZM*21] GAO, LIN, ZHANG, LING-XIAO, MENG, HSIEN-YU, et al. "PRS-Net: Planar Reflective Symmetry Detection Net for 3D Models". *IEEE Trans. on Vis. and Computer Graph.* 27.6 (2021), 3007–3018 1, 2.

[HKLM22] HRUDA, LUKÁŠ, KOLINGEROVÁ, IVANA, LÁVIČKA, MIROSLAV, and MAŇÁK, MARTIN. "Rotational symmetry detection in 3D using reflectional symmetry candidates and quaternion-based rotation parameterization". *Computer Aided Geometric Design* 98 (2022), 102138 1.

[LOW*20] LIU, LI, OUYANG, WANLI, WANG, XIAOGANG, et al. "Deep Learning for Generic Object Detection: A Survey". *Int. J. Comput. Vision* 128.2 (Feb. 2020), 261–318 3.

[NR20] NAGAR, RAJENDRA and RAMAN, SHANMUGANATHAN. "3DSymm: Robust and Accurate 3D Reflection Symmetry Detection". *Pattern Recognition* 107 (2020), 107483 1.

[PWT*22] PANG, YATIAN, WANG, WENXIAO, TAY, FRANCIS EH, et al. "Masked autoencoders for point cloud self-supervised learning". *Computer Vision–ECCV 2022: 17th European Conf., Tel Aviv, Israel, October 23–27, 2022, Proc., Part II*. Springer. 2022, 604–621 4.

[QYSG17] QI, CHARLES R, YI, LI, SU, HAO, and GUIBAS, LEONIDAS J. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". *arXiv preprint arXiv:1706.02413* (2017) 4.

[RRB*22] ROMANENGO, CHIARA, RAFFO, ANDREA, BIASOTTI, SILVIA, et al. "SHREC 2022: Fitting and recognition of simple geometric primitives on point clouds". *Computers & Graphics* 107 (2022), 32–49 2.

[SC20] SHARP, NICHOLAS and CRANE, KEENAN. "A Laplacian for Non-manifold Triangle Meshes". *Computer Graphics Forum (SGP)* 39.5 (2020) 5.

[SGS14] SIPIRAN, IVAN, GREGOR, ROBERT, and SCHRECK, TOBIAS. "Approximate Symmetry Detection in Partial 3D Meshes". *Computer Graphics Forum* 33.7 (2014), 131–140 1, 5.

[Shi95] SHIKIN, EUGENE V. *Handbook and atlas of curves*. CRC Press, 1995 2.

[SHZ*20a] SHI, YIFEI, HUANG, JUNWEN, ZHANG, HONGJIA, et al. "SymmetryNet: Learning to Predict Reflectional and Rotational Symmetries of 3D Shapes from Single-View RGB-D Images". *ACM Trans. Graph.* 39.6 (Nov. 2020) 1, 2.

[SHZ*20b] SHI, YIFEI, HUANG, JUNWEN, ZHANG, HONGJIA, et al. "SymmetryNet: Learning to Predict Reflectional and Rotational Symmetries of 3D Shapes from Single-View RGB-D Images". *ACM Trans. Graph.* 39 (2020) 4, 6.

[SKKC22] SEO, AHYUN, KIM, BYUNGJIN, KWAK, SUHA, and CHO, MINSU. "Reflection and Rotation Symmetry Detection via Equivariant Learning". *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. June 2022, 9539–9548 1.

[SWZ*18] SUN, XINGYUAN, WU, JIAJUN, ZHANG, XIUMING, et al. "Pix3d: Dataset and methods for single-image 3d shape modeling". *Proc. of the IEEE Conf. on computer vision and pattern recognition*. 2018, 2974–2983 1.

[TK12] TSOGKAS, STAVROS and KOKKINOS, IASONAS. "Learning-Based Symmetry Detection in Natural Images". *Computer Vision – ECCV 2012*. Ed. by FITZGIBBON, ANDREW, LAZEBNIK, SVETLANA, PERONA, PIETRO, et al. Springer Berlin Heidelberg, 2012, 41–54 1.

[WSK*15] WU, ZHIRONG, SONG, S., KHOSLA, A., et al. "3D ShapeNets: A deep representation for volumetric shapes". *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2015, 1912–1920 2.