PRS-Net: Planar Reflective Symmetry Detection Net for 3D Models

Lin Gao, Ling-Xiao Zhang, Hsien-Yu Meng, Yi-Hui Ren, Yu-Kun Lai, Leif Kobbelt

Abstract—In geometry processing, symmetry is a universal type of high-level structural information of 3D models and benefits many geometry processing tasks including shape segmentation, alignment, matching, and completion. Thus it is an important problem to analyze various symmetry forms of 3D shapes. Planar reflective symmetry is the most fundamental one. Traditional methods based on spatial sampling can be time-consuming and may not be able to identify all the symmetry planes. In this paper, we present a novel learning framework to automatically discover global planar reflective symmetry of a 3D shape. Our framework trains an unsupervised 3D convolutional neural network to extract global model features and then outputs possible global symmetry parameters, where input shapes are represented using voxels. We introduce a dedicated symmetry distance loss along with a regularization loss to avoid generating duplicated symmetry planes. Our network can also identify generalized cylinders by predicting their rotation axes. We further provide a method to remove invalid and duplicated planes and axes. We demonstrate that our method is able to produce reliable and accurate results. Our neural network based method is hundreds of times faster than the state-of-the-art methods, which are based on sampling. Our method is also robust even with noisy or incomplete input surfaces.

Index Terms—Deep Learning, Symmetry Detection, 3D Models, Planar Reflective Symmetry

1 Introduction

The vast majority of species and man-made objects exhibit symmetrical patterns. For example, bilateral symmetry and radial symmetry are two common patterns existing in starfish and sunflower shapes. Symmetry is also an important concept in mathematics. An object is symmetric if some properties do not change under certain transformations. In geometric processing, finding symmetries in geometric data, such as point clouds, polygon meshes and voxels, is an important problem, because numerous applications take advantage of symmetry information to solve their tasks or improve the algorithms, *e.g.*, shape matching [1], segmentation [2], completion [3], *etc.*

Detecting symmetry of 3D objects is an essential step for many applications. Among all the symmetry types, the most common and important one is planar reflective symmetry. In a simple case, a shape can be aligned to the principal axes by applying principal component analysis (PCA). Then planes formed by pairs of principal axes (*i.e.*, orthogonal to the remaining principal axis) can be checked to see if they are symmetry planes. This simple approach works for simpler cases, where the symmetry plane is well aligned with principal axes. However, it is unable to detect symmetry planes which are not orthogonal to any principal axis (*e.g.*, any plane that passes through the rotation axis of a cylinder is a symmetry plane which may not be orthogonal to any principal axis). Moreover, the method is highly sensitive to even small changes of geometry, leading to poor detection results. The state-of-the-art methods for symmetry plane detection [4], [5] are based on spatial sampling which is much more robust than PCA. However, since they need to sample many potential candidates, the output may produce poor or inaccurate results depending on the random sampling.

To address such limitations, in this paper, we introduce a novel learning framework to automatically discover global planar reflective symmetry. We use a deep convolutional neural network (CNN) to extract the global model feature and capture possible global symmetry. To make CNN-based learning easier and more effective, we convert shapes in arbitrary representations to voxels, before feeding them into our network. The output of our network involves parameters representing reflection planes and rotation axes. Although our aim is to detect planar reflective symmetry, for surfaces of revolution such as a cylinder, any plane passing through the rotation axis can be a symmetry plane. By detecting rotation axes explicitly, we can ensure such symmetry planes are fully detected. Our network is unsupervised because we do not require any annotations for symmetry planes of the target objects. This makes collecting training data much easier. To achieve this, we introduce a novel symmetry distance loss and a regularization loss to effectively train our network. The former measures deviation of the geometry from symmetry given a potential symmetry plane, and the latter is used to avoid generating duplicated symmetry planes. We further provide a post-processing method to remove invalid and duplicated planes and axes. Compared with the method of Kazhdan *et al.* [4], our method can produce more reliable and accurate results. More importantly, our learning based approach is hundreds of times faster, achieving real-time performance. We also

- *L. Gao, L.X. Zhang and Y.H. Ren are with Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. E-mail:{gaolin, zhanglingxiao}@ict.ac.cn, renyihui17@mails.ucas.ac.cn*
- *H.Y. Meng is with University of Maryland, College Park, Maryland, US. E-mail:mengxy19@cs.umd.edu*
- *Y.K. Lai is with Visual Computing Group, School of Computer Science and Informatics, Cardiff University, Wales, UK. E-mail:LaiY4@cardiff.ac.uk*
- *L. Kobbelt is with Institute for Computer Graphics and Multimedia, RWTH Aachen University, Aachen, Germany. E-mail:kobbelt@cs.rwth-aachen.de*

Manuscript received April 19, 2019; revised August 26, 2019.

show our network is robust even for noisy and incomplete input. Our key contributions are:

- We develop PRS-Net, the first unsupervised deep learning approach to detecting global planar reflective symmetry of 3D objects. Our approach is hundreds of times faster than state-of-the-art methods and also more accurate and reliable.
- We model the symmetry detection problem as a differentiable function, which can be attained by a neural network. We further design a dedicated symmetry distance loss along with a regularization loss to avoid generating duplicated symmetry planes. Thanks to the loss functions, our network is trained in an unsupervised manner, making data collection much easier.

2 Related Work

2.1 Symmetry Detection

Symmetry detection is an important topic in shape analysis, and is widely used in 2D images and 3D geometry. Symmetry detection includes global and partial symmetry, as well as intrinsic and extrinsic symmetry. Most methods can cope with certain level of approximate symmetry.

For 2D image symmetry detection, traditional methods usually vote on the parameters of the symmetry by matched points and angles. Marola [6] introduces a metric named coefficient of symmetry for finding symmetry axes of 2D images. Sun and Si [7] use Fourier analysis to find symmetry axes of images. Loy and Eklundh [8] introduce a simple method for grouping symmetric constellations of features. Some methods [9], [10] detect curved reflection symmetry. Hauagge and Snavely [11] propose a method to extract multi-scale local bilateral and rotational symmetries. There are also some pioneer research works for 2D symmetry detection based on neural networks. Zielke *et al.* [12] introduce a feedforward network named SEED to find the contours of symmetric objects. Fukushima and Kikuchi [13] propose a method to find symmetry axes using a neural network. Funk and Liu [14] introduce the Sym-Net to detect reflection and rotational symmetry. Vasudevan *et al.* [15] handle symmetry classification using a deep convolutional neural network in Fourier space. These methods focus on 2D image symmetry detection, rather than symmetry on 3D shapes.

For symmetry detection of 3D objects, Atallah [16] proposes an algorithm for enumerating all the axes of symmetry of a planar figure consisting of simple components such as segments, circles and points. Martinet *et al.* [17] propose a method for detecting global accurate symmetry using generalized moments. Kazhdan *et al.* [4] detect n-fold rotational symmetry based on the correlation of the spherical harmonic coefficients. Raviv *et al.* [18] present a generalization of symmetries for non-rigid shapes.

Based on whether the symmetry exists in the (Euclidean) embedding space, or based on distance metrics of the geometry, symmetry can be classified as extrinsic and intrinsic.

For extrinsic symmetry, we usually use the Euclidean distance between points to measure the symmetry of a shape, while intrinsic symmetry is measured by different metrics. For global extrinsic symmetry, planar reflective symmetry is the most fundamental one. Zabrodsky *et al.* [19] introduce a measure of approximate symmetry. Podolak *et al.* [2] further describe a planar reflective symmetry transform (PRST) that captures a continuous measure to help find the reflective symmetry. Li *et al.* [20] present a detection method based on the viewpoint entropy features. Ecins *et al.* [21] introduce a method that mainly detects symmetric objects in 3D scenes and scans of real environment. Cicconet *et al.* [22] regard the problem of finding planar symmetry as a problem of registering two datasets. Ji and Liu [23] propose a network to detect reflection symmetry, but their method needs supervised learning with annotated data. In contrast, our method is unsupervised, without requiring annotated data for training.

For intrinsic symmetry detection, Ovsjanikov *et al.* [24] introduce a method to compute intrinsic symmetry of a shape using eigenfunctions of Laplace-Beltrami operators. Kim *et al.* [25] present a method to discover point correspondences to detect global intrinsic symmetry on 3D models based on the algorithm by Lipman *et al.* [26]. Mitra *et al.* [27] present a method to detect intrinsic regularity, where the repetitions are on the intrinsic grid.

For partial symmetry detection, a shape S is said to have partial symmetry w.r.t. a transformation T, if there are two subsets $S_1, S_2 \subset S$ such that $T(S_1) = S_2$. Gal and Cohen-Or [28] introduce local surface descriptors that represent the geometry of local regions of the surface to detect partial symmetry. Mitra *et al.* [29] present a method based on transformation space voting schemes to detect partial and approximate symmetry. Pauly *et al.* [30] present a method for discovering regular or repeated geometric structures in 3D shapes. Berner *et al.* [31] present a symmetry detection algorithm based on analyzing a graph of surface features. Lipman *et al.* [32] introduce the Symmetry Factored Embedding (SFE) and the Symmetry Factored Distance (SFD) to analyze and represent symmetries in a point set. Xu *et al.* [33] extend *PRST* [2] to extract partial intrinsic reflective symmetries.

Our aim is to develop an unsupervised deep learning approach for effective *real-time* global planar reflective symmetry detection.

2.2 Geometry Processing with Deep Learning

Neural networks have achieved much success in various areas. In recent years, more and more researchers generalize this tool from 2D images to 3D geometry. Su *et al.* [34] use a dimensionality reduction strategy that puts 2D rendered images of a 3D object from multiple views into several classical and mature 2D CNNs. Maturana [35] argues that many existing systems do not take full advantage of 3D depth information, so they create a volumetric occupancy grid representation and predict 3D targets in real time directly from the 3D CNN. Wu *et al.* [36] introduce 3D ShapeNet to learn the distribution of complex 3D voxel grids and use it for 3D shape classification and recognition. Girdhar *et al.* [37] propose a TL-embedding network to generate 3D voxel models from 2D images. Qi *et al.* [38] combine a volumetric CNN with a multi-view CNN, enabling it to be used for object classification of 3D data. Wu *et al.* [39] propose 3D-GAN (Generative Adversarial Network) to generate 3D

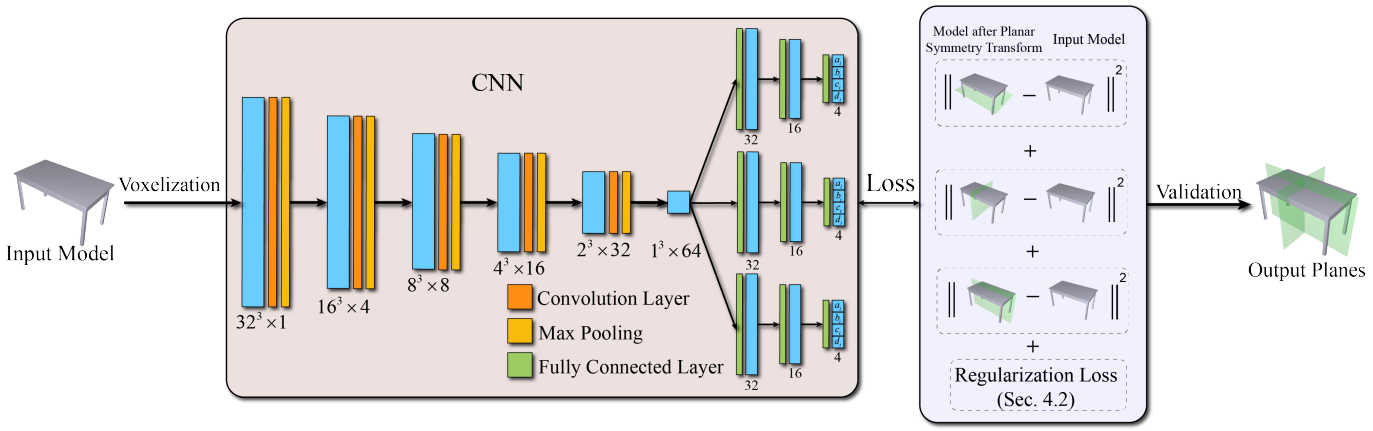


Fig. 1. Overview of our method. The input to the network is the voxelized volume of the mesh model. A CNN is used to predict the parameters of the symmetry planes and the planar reflective symmetry transforms associated with the symmetry planes define the loss to train the CNN. This makes the training of CNN operate in an unsupervised manner without any labeled data. The regularization loss is used to avoid predicting repetitive symmetry planes. To simplify the network architecture, our network predicts a fixed number (three in practice) of symmetry planes and rotation axes, which may not all be valid. Duplicated or invalid symmetry planes are removed in the validation stage.

objects from a probabilistic space. Tulsiani *et al.* [40] present a network to interpret 3D objects with a set of volumetric primitives. Riegler *et al.* [41] present OctNet for 3D object classification, using an octree structure to reduce memory and computational costs. Liu *et al.* [42] present a network for piece-wise planar reconstruction from a single RGB image.

For point cloud representation, Qi *et al.* [43] introduce PointNet for classification and segmentation. Engelmann *et al.* [44] build a network based on PointNet to deal with large-scale scene semantic segmentation. Wang *et al.* [45] propose a new module named EdgeConv, which acts on dynamic graphs for CNNs, and encodes local neighborhood information. Wang and Solomon [46] propose a learning-based method named Deep Closest Point (DCP) to predict rigid transformation for point cloud registration, and a Partial Registration Network (PRNet) [47] for partial-to-partial registration. Meng *et al.* [48] propose a network which applies group convolutions on regular voxel grids and encodes features computed from radial basis functions (RBF), for point cloud segmentation, achieving state-of-the-art results. Besides, there are some methods that deal with graphs and mesh representations. Hanocka *et al.* [49] propose MeshCNN, which defines mesh convolution and pooling on the mesh edges to handle various tasks including mesh classification and semantic segmentation. Gao *et al.* [50] introduce VAE-Cycle-GAN for automatic unpaired shape deformation transfer. Some methods analyze man-made objects using tree or graph structures to handle model reconstruction, generation and interpolation, such as [51], [52], [53].

Such works show great potential for using deep learning for 3D geometry processing, but none of the existing work considers learning to detect 3D object symmetry in an unsupervised manner, which we will address in this paper.

3 Network Architecture

In this section, we describe the network architecture of our method. The overall network is presented in Figure 1. This work aims to train a CNN to predict the symmetry planes in an unsupervised manner. The CNN has five 3D convolution layers of kernel 3, padding 1, and stride 1. After each 3D convolution, a max pooling of stride 2 and leaky ReLU [54] activation are applied. These are followed by fully connected layers to predict the parameters of symmetry planes.

The input of the network is $32 \times 32 \times 32$ voxels which are voxelized from the input shape. As we will later show in Section 6.7, this resolution achieves better performance than alternative settings.

The 3D convolution and pooling are used to extract global features of the shape. The output includes parameters of reflective planes and rotation parameters. For typical shapes, our network predicts three potential symmetry planes and three potential rotation axes. These will be further validated in the validation stage so the shape may have fewer (or even none) symmetry planes. The symmetry planes $\mathbf{P}_i$ $(i = 1, 2, 3)$ are represented using an implicit representation. We further use quaternions to represent rotations $\mathbf{R}_i$ $(i = 1, 2, 3)$, because the quaternion is more compact with fewer parameters compared to the rotation matrix, and it can be easily transformed from and to an axis-angle representation. We initialize the normal vectors of the planes and the directions of the axes to be three vectors perpendicular to each other to maximize their coverage. In practice, we simply set $\mathbf{v_1} = (1, 0, 0), \mathbf{v_2} = (0, 1, 0), \mathbf{v_3} = (0, 0, 1)$ to initialize them. The initial angle of each rotation axis is set to $\theta = \pi$, thus the corresponding quaternion is $\mathbf{R}_i = \cos(\theta/2) + \sin(\theta/2)(\mathbf{v}_i^{(1)}\mathbf{i} + \mathbf{v}_i^{(2)}\mathbf{j} + \mathbf{v}_i^{(3)}\mathbf{k})$, where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the fundamental quaternion units, θ is the rotation angle, and $\mathbf{v}_i^{(k)}$ is the k^{th} component of $\mathbf{v}_i$ $(k = 1, 2, 3$ corresponding to x, y and $z)$. In our network, the predicted quaternions $\mathbf{R_i}$ are normalized to a unit vector after each iteration of optimization. After the training is finished, we transform the quaternion representation to the axis-angle representation.

Our network is trained in an unsupervised manner because we do not require any annotations for the reflection plane parameters that best describe the global extrinsic symmetry of the object. This greatly reduces the effort of

obtaining training data as only a collection of (symmetric) shapes is required. In order to achieve this, we propose a novel symmetry distance loss to promote planar symmetry. Moreover, to avoid producing duplicated symmetry planes, we further introduce a regularization loss.

4 Loss Function

Denote by $\mathbf{P}_i = (\mathbf{n}_i, d_i)$ the i^{th} plane parameters in implicit form, where $\mathbf{n}_i = (a_i, b_i, c_i)$ is the normal direction of the plane, which uniquely determines a reflection plane $a_i x + b_i y + c_i z + d_i = 0$, and $\mathbf{R}_i = (u_{i0}, u_{i1}, u_{i2}, u_{i3})$ is the i^{th} rotation parameters, which represent the quaternion $u_{i0} + u_{i1}\mathbf{i} + u_{i2}\mathbf{j} + u_{i3}\mathbf{k}$ of rotation transform. To train the network to predict symmetry planes and rotation axes, we introduce two loss functions, namely symmetry distance loss and regularization loss.

4.1 Symmetry Distance Loss

To measure whether an input shape O is symmetric w.r.t. a given reflection plane or a rotation axis, we first uniformly sample N points on the shape to form a point set $\mathcal{Q}$. We then obtain a transformed point set $\mathcal{Q}'$ by applying planar symmetry or rotation transformation to each point $\mathbf{q}_k \in \mathcal{Q}$ to obtain the transformed point $\mathbf{q}'_k$.

For the i^{th} reflection plane, the symmetry point $\mathbf{q}'_k$ of point $\mathbf{q}_k$ is:

$$\mathbf{q}'_k = \mathbf{q}_k - 2\frac{\mathbf{q}_k \cdot \mathbf{n}_i + d_i}{\|\mathbf{n}_i\|^2}\mathbf{n}_i, \tag{1}$$

where $\mathbf{n}_i$ is the normal vector of the i^{th} reflection plane, and d_i is its offset parameter.

For the j^{th} rotation axis, the symmetry point $\mathbf{q}'_k$ of point $\mathbf{q}_k$ is

$$\hat{\mathbf{q}}'_k = \mathbf{p}_j \hat{\mathbf{q}}_k \mathbf{p}_j^{-1}, \tag{2}$$

where $\mathbf{p}_j = (p_{j0}, p_{j1}, p_{j2}, p_{j3})$ represents the quaternion of the j^{th} rotation, and $\hat{\mathbf{q}}_k = (0, \mathbf{q}_k)$ is the quaternion form of point $\mathbf{q}_k$. This results in a new quaternion $\hat{\mathbf{q}}'_k$, with $\mathbf{q}'_k$ as its imaginary part.

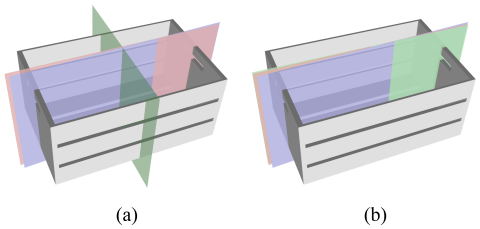
Then we calculate the shortest distance

$$D_k = \min_{\mathbf{o} \in O} \|\mathbf{q}'_k - \mathbf{o}\|_2 \tag{3}$$

from symmetry points $\mathbf{q}'_k$ to the shape O. To calculate D_k efficiently, we precompute the closest point on the surface to each grid center point of a regular grid, and during training, we calculate the distance between symmetry points to the corresponding closest point in the same grid as the approximate closest distance and their gradients required for back propagation. Finally, the symmetry distance loss of a shape is defined as

$$L_{sd} = \sum_{i=1}^{3}\sum_{k=1}^{N} \hat{D}_k^{(i)} + \sum_{j=1}^{3}\sum_{k=1}^{N} \tilde{D}_k^{(j)} \tag{4}$$

where $N = |\mathcal{Q}|$ is the number of sample points, $\hat{D}_k^{(i)}$ and $\tilde{D}_k^{(j)}$ represent the symmetry error for the k^{th} sample point w.r.t. the i^{th} symmetry plane and the j^{th} rotation axis, respectively, as defined in Eq. 3.



(a) (b)

Fig. 2. The results of the network with (a) and without (b) the regularization loss. We train the network with the same initial plane parameters. Without the regularization loss, the network cannot differentiate near identical planes, as they also achieve a local minimum of the loss function during training. Our regularization loss effectively promotes non-identical outputs, which helps to cover different symmetry planes.

4.2 Regularization Loss

Many shapes have multiple symmetry planes or rotation axes. Since each of them is sufficient to minimize the symmetry distance loss, the network may produce multiple, near-identical outputs (*e.g.* $\mathbf{P}_1 \approx \mathbf{P}_2$). This may lead to an output that misses essential symmetry planes/rotation axes. To address this, we aim to constrain the learning of reflection planes and rotation axes to not overlap with each other, by adding a regularization loss L_r to separate each plane and axis from each other as much as possible. Let $\mathbf{M}_1$ be a 3×3 matrix where each row is the unit normal direction of a symmetry plane, *i.e.*, $\mathbf{M}_1 = [\frac{\mathbf{n}_1}{\|\mathbf{n}_1\|} \quad \frac{\mathbf{n}_2}{\|\mathbf{n}_2\|} \quad \frac{\mathbf{n}_3}{\|\mathbf{n}_3\|}]^T$. Let $\mathbf{M}_2$ be another 3×3 matrix where each row contains the normalized axis direction of rotational symmetry prediction. Let

$$\mathbf{A} = \mathbf{M}_1\mathbf{M}_1^T - \mathbf{I}, \tag{5}$$

$$\mathbf{B} = \mathbf{M}_2\mathbf{M}_2^T - \mathbf{I}, \tag{6}$$

where $\mathbf{I}$ is the 3×3 identity matrix. If each plane (resp. axis) is orthogonal to every other plane (resp. axis), then $\mathbf{A}$ and $\mathbf{B}$ are all-zero matrices. We define the regularization loss as

$$L_r = \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2 = \sum_{i=1}^{3}\sum_{j=1}^{3}(\mathbf{A}_{ij}^2 + \mathbf{B}_{ij}^2), \tag{7}$$

which penalizes planes and axes closer to parallel, where $\|\cdot\|_F$ is the Frobenius norm. Figure 2 compares the results with (a) and without (b) the regularization loss. This regularization term is the soft constraint and would not enforce the planes to be strictly perpendicular, as shown in Figure 8. In this experiment, we initialize all the planes and axes with the same settings. As can be seen, the reflection planes overlap with each other without the regularization loss as shown in the right column, because it is difficult for the network to separate them as they also achieve the same local minimum, while the planes are clearly separated on the left thanks to the regularization. There are still two overlapping planes because the model does not have the third reflection plane with small symmetry distance loss (which will be addressed in validation).

4.3 Overall Loss Function

We define the overall loss function as

$$L = L_{sd} + w_r L_r, \tag{8}$$

where w_r is a weight to balance the importance of two loss terms.

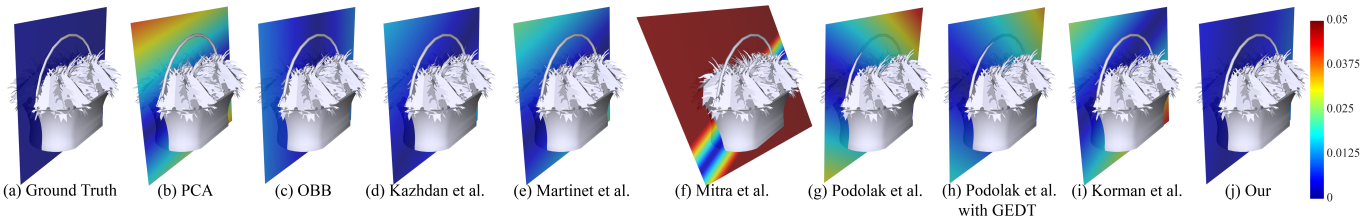


Fig. 3. Comparison of symmetry plane detection using different methods. The heatmap describes the distance between the ground truth and the plane of each correspond method. It shows that our method has less error and more reasonable results than other methods.

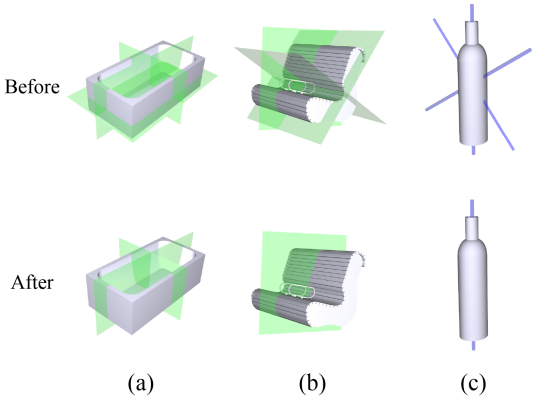


Fig. 4. The results before (the first row) and after (the second row) validation. Most models have fewer than 3 symmetry planes/rotation axes, and the initial network output may contain extra symmetry outputs (which are local minima or duplicates). Our simple validation stage removes duplicated outputs as well as symmetry outputs which are not of acceptable quality.

5 Validation

Our network predicts symmetry planes/rotation axes and it always predicts three symmetry planes and three rotations to simplify the architecture. However, real-world shapes may have fewer symmetry planes and rotation axes. In this case, some output planes may overlap with each other. Moreover, due to the local minimum nature of gradient descent optimization, the network may also detect some approximate symmetry which is not sufficiently good. These issues however can be easily addressed by a simple validation stage. We check the detected symmetry planes and rotation axes to remove duplicated outputs: if its dihedral angle is less than $\pi/6$, we remove the one with larger symmetry distance error. Meanwhile, if the detected symmetry planes/rotation axes lead to high symmetry distance loss (greater than 4×10^{-4} in our experiments), we also remove them as they are not sufficiently symmetric. In particular for rotational symmetry, as we are only concerned with detecting surfaces of revolution with infinite symmetry planes going through the symmetry axis, for each detected rotation axis, we consider rotations by an arbitrary angle (every 1° in practice), and only accept it if the rotational symmetry is retained with arbitrary rotation angles.

Note that we normalize the shapes to fit in a unit cube, so this threshold is generally applicable, and obtained by working out the distributions of symmetry distance errors for a small number of valid cases. By normalizing the shapes to the unit cube, this threshold is applicable for all the shapes. As shown in Figure 4(a), the bath only has two reflection planes, but the network always outputs three symmetry planes before validation. Their symmetry distance errors are $6.67 \times 10^{-5}, 9.43 \times 10^{-5}$ and 2.57×10^{-3} respectively, so the validation removes the third plane and retains the other two planes. The symmetry distance errors of the three output planes of the bench in Figure 4(b) are 9.50×10^{-6}, 1.46×10^{-3} and 1.31×10^{-3}, so two planes are removed due to high symmetry distance loss. Similarly, two extra rotation axes are removed in Figure 4(c).

6 Experiments

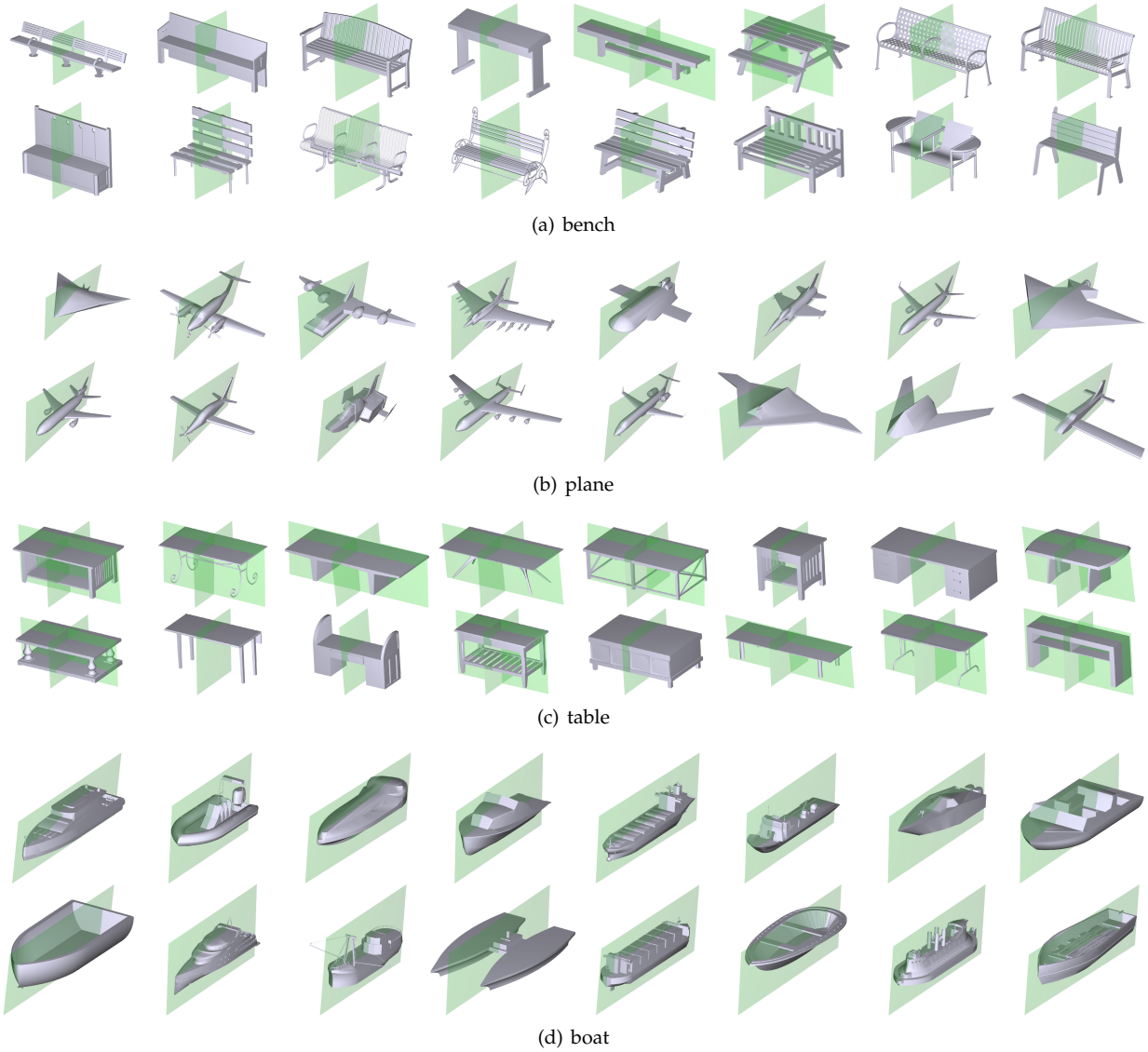
6.1 Dataset and Training

We use the ShapeNet [55] dataset which contains 55 common object categories with about $51,300$ unique 3D models to train our network. We choose 80% models as the training set and the remaining 20% models for testing. Because the ShapeNet models are often roughly axis aligned, and many categories have fewer than 500 models, we apply different random rotations on each model to obtain $4,000$ augmented models for each category for training. However, many shapes in ShapeNet are not symmetric. For non-symmetric shapes, it can be difficult to determine which result is better even by human annotators. Although *e.g.* symmetry distance error (SDE) that measures how close the shape is to itself after mirroring could be used to give an indication, it does not provide a definite measure. We are not aware of benchmark datasets for symmetry, and thus we collect a test set containing shapes which are manually validated to ensure they are symmetric, and for these shapes, we obtain reasonable ground truth symmetry planes to compare the errors of different methods w.r.t. the ground truth. Because the models in ShapeNet are pre-aligned, we select three axis-aligned planes to be candidates and manually check and correct the results to get the valid planes for each model in the test set as the ground truth. For models with more than three symmetry planes, we also add these planes as the ground truth.

In the preprocessing step, we uniformly sample $1,000$ points on the surface, and generate voxels of size $32 \times 32 \times 32$ as input to our network. During network training, we set batch size $b = 32$, learning rate $l_r = 0.01$ and regularization loss weight $w_r = 25$, and then use the ADAM [56] optimizer to train our network according to the loss described above.

6.2 Results and Evaluations

We compare our method with PCA, Oriented Bounding Box [57], the methods of Kazhdan *et al.* [4], Martinet *et al.* [17], Mitra *et al.* [29], Podolak *et al.* [2], Podolak *et al.* [2] with Gaussian Euclidean Distance Transform (GEDT), using their default parameters, and Korman *et al.* [5]. To quantitatively evaluate the quality of detected symmetry planes, we first



(a) bench

(b) plane

(c) table

(d) boat

Fig. 5. More results of symmetry planes discovered by our method of different shapes from test data in ShapeNet [55], including (a) bench, (b) plane, (c) table, and (d) boat.

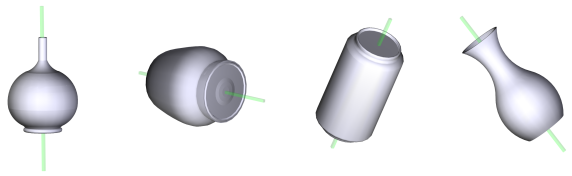


Fig. 6. The rotation axes of generalized cylinders detected by our method. Our method can identify these shapes through outputting quaternions that represent rotational symmetry.

normalize the normal vector of the plane, and adjust the direction of the normal vector to make the angle between the detected normal and ground truth normal no larger than $\pi/2$, and the error of the plane w.r.t. the ground truth (GTE) is defined as

$$GTE = (a_i - a_{gt})^2 + (b_i - b_{gt})^2 + (c_i - c_{gt})^2 + (d_i - d_{gt})^2. \quad (9)$$

Alternatively, we can also use symmetry distance error (SDE) defined in the same way as L_{sd} to measure the symmetry quality.

We compare our method with existing methods. Figure 3 shows the results of reflection planes from different methods. In each figure, the heatmap presents the distance between the sample points on the ground truth plane and the reflection points on each result plane. The methods of Kazhdan *et al.* [4] and PRST [2] are sensitive to the resolution of grids and the distribution of sample points. For PRST, it may also produce different results when it runs multiple times due to the random sampling. The computational time of the method grows quickly when more sample points and higher resolution of grids are used. The last column shows that our method obtains the most accurate result. Moreover, our method only takes a few milliseconds, once the network is trained, while methods except PCA need several iterations to compute the local optimal result. More results as shown in Figure 5, our method is able to produce reliable and accurate results, including shapes with multiple symmetry planes. In Figure 6, we show the rotation axes of the generalized cylinders detected by our method. To

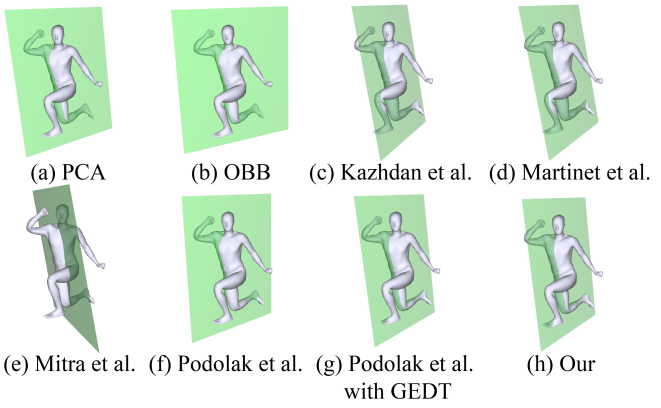


Fig. 7. Comparison of symmetry detection for a human shape from the SCAPE dataset [60] only with approximate symmetry using different methods. Our result has the lowest symmetry distance error.

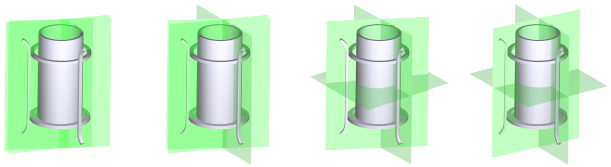


Fig. 8. An example of visual results with different regularization weights w_r. The table shape has two symmetry planes, and planes overlap with each other when the regularization loss weight w_r is 1 or 10, as shown in (a) and (b). The network produces near-identical outputs because the regularization loss with such small weights has little impact and the algorithm identifies duplicated planes so fails to identify the other symmetry plane. In (c) and (d), the planes are separated with large regularization weights, but (c) produces more accurate symmetry planes than (d).

detect these shapes, we check the symmetry distance error of rotation axes.

We also evaluate our network on large test sets. Table 1 shows the mean ground truth error and symmetry distance error of our test set with 1000 models, and our method produces minimum error.

Many shapes are not entirely symmetric, so we evaluate our method on human shapes from the SCAPE dataset [60] to test the capability of handling such general cases. As shown in Figure 7, we visualize the symmetry plane with lowest symmetry distance error of each method except for the method of Korman *et al.* [5] which fails to detect any reflective symmetry, and our method discovers a more plausible symmetry plane, which has the lowest symmetry distance error 1.75×10^{-3}, compared with 3.49×10^{-3} of PCA, 3.98×10^{-3} of Oriented Bounding Box, 1.83×10^{-3} of the method of Kazhdan *et al.* [4], 1.90×10^{-3} of the method of Martinet *et al.* [17], 4.87×10^{-3} of the method of Mitra *et al.* [29], 3.78×10^{-3} of PRST [2] and 2.01×10^{-3} of PRST with GEDT. It demonstrates that our method has the ability to detect *approximate* symmetry planes of general shapes, outperforming existing methods. This is because we use the unsupervised loss to train the network, and the ShapeNet dataset [55] has various categories including symmetric and asymmetric shapes. Note that our training set does not include any shape from SCAPE or even any human shape. This also shows that our network generalizes well to new shapes and unseen shape categories.

In Table 2, we also report accuracy comparison of our method with alternative methods (in terms of SDE as no ground truth is available) on ABC [58] and Thingi10K [59] datasets, which contain a large number of asymmetric shapes. In this experiment, we randomly select 80% data for training and 20% for testing, and use the training data to fine-tune the network pre-trained on ShapeNet. Our method performs the best on both ABC [58] and Thingi10K [59]. This demonstrates that our method generalizes well to new datasets, producing minimum average SDE and dealing well with more complex and asymmetric shapes.

6.3 Computation Time

We calculate the computation time and compare it with alternative methods. Our experiments were carried out on a desktop computer with a 3.6 GHz Intel Core i7-6850K CPU, 128G memory and an NVIDIA TITAN X GPU. For a typical model, such as the piano in Figure 11 with 1,052 vertices and 4,532 faces, OBB [57], the method of Kazhdan *et al.* [4], the method of Martinet *et al.* [17], the method of Podolak *et al.* [2], the method of Podolak *et al.* [2] with GEDT, the method of Korman *et al.* [5] and the method of Mitra *et al.* [29] require 0.02, 0.51, 2.82, 3.40, 5.00, 0.97, 0.42 seconds, whereas our method only needs 1.81 ms, which is hundreds faster than the state-of-the-art methods and achieves real-time performance. This is because these methods use sampling and/or iterative algorithms which increases the computation time. In terms of running times, this method is also comparable with PCA (typically taking 1.9 ms using CPU) since the trained network is performed on the GPU with powerful computation ability.

6.4 Choice of parameter w_r

We show an example of visual results with different regularization loss weight in Figure 8. We choose $w_r = 25$ for training in our paper because it separates multiple symmetry planes properly, unlike Figure 8(a) and Figure 8(b) with small w_r, and has lower symmetry distance loss than Figure 8(d).

6.5 Application to Shape Completion

With the predicted symmetry planes, many geometric tasks would benefit. Here, we apply this to the application of shape completion. We show a comparison of shape completion in Figure 9. We visualize the Euclidean error between the ground truth part and the generated part of the piano, which is obtained by mirroring the geometry of the left leg along the symmetry plane detected by different methods. Our method and OBB give better symmetry planes than other methods, and therefore produce good completed geometry. The method of Martinet *et al.* [17] produces worse result because it is designed to detect accurate symmetry. The method of Mitra *et al.* [29] and Podolak *et al.* [2] use sampling algorithms to get the reflection planes which is affected by the partial shape. OBB is hardly affected by such missing parts, and our network learns the global shapes of the model through 3D convolutions and numerous training data, and discovers the global symmetry reliably. Thanks to this symmetry detection method, the incomplete shapes with reflective symmetries can be repaired accurately and efficiently.

TABLE 1
The mean ground truth error (GTE) and the symmetry distance error (SDE) of different methods. The reported results are averaged over a validated subset of ShapeNet with 1000 models from different categories. Our method produces minimum average errors for both SDE and GTE.

Error	PCA	Oriented Bounding Box [57]	Kazhdan *et.al.* [4]	Martinet *et.al.* [17]	Mitra *et.al.* [29]	PRST [2]	PRST [2] with GEDT	Korman *et.al.* [5]	Our
GTE($\times 10^{-2}$)	2.41	1.24	0.17	13.6	52.1	4.42	3.97	19.2	**0.11**
SDE($\times 10^{-4}$)	3.32	1.25	0.897	3.95	14.2	1.78	1.60	1.75	**0.861**

TABLE 2
The symmetry distance error ($\times 10^{-4}$) of different methods on ABC [58] and Thingi10K [59]. Our method produces minimum average SDE and can deal with more complex and asymmetric shapes.

Dataset	PCA	Oriented Bounding Box [57]	Kazhdan *et.al.* [4]	Martinet *et.al.* [17]	Mitra *et.al.* [29]	PRST [2]	PRST [2] with GEDT	Korman *et.al.* [5]	Our
ABC [58]	6.97	4.98	5.54	7.48	12.1	6.28	4.87	4.54	**1.14**
Thingi10K [59]	7.24	2.78	4.34	4.37	12.0	5.43	4.97	2.11	**1.69**

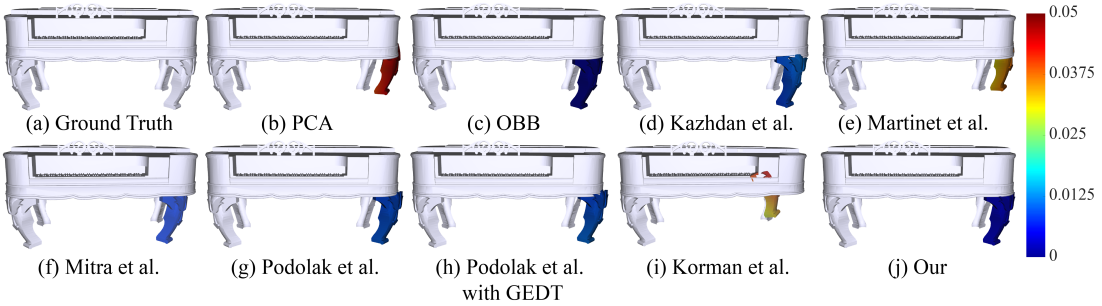


Fig. 9. Comparison of shape completion. Our method repairs the incomplete shape perfectly due to the robustness and accuracy of our method. It shows the Euclidean error between the ground truth part and the generated part of the piano, which is obtained by mirroring the geometry of the left leg along the symmetry planes detected by different methods.

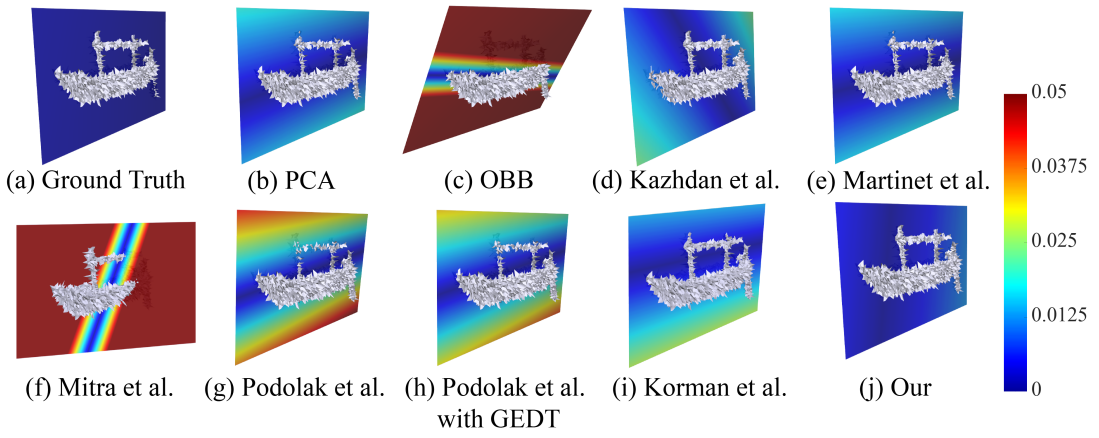


Fig. 10. Robustness testing with adding Gaussian noise to each vertex along the normal direction. It shows that our method produces stable output.

6.6 Robustness

In order to test the robustness of our network, we present two different experiments, including noisy and incomplete models. These experiments are motivated by the fact that scanned models often contain noisy and/or incomplete surfaces. As Figure 10 shows, because most ShapeNet models are non-manifold and some of them have fewer than 1000 vertices which result in poor noisy models, we first use the method of Huang *et al.* [61] to convert the model to a manifold, before adding Gaussian noise on each vertex along the normal direction. It shows that our method produces stable output with smallest error, demonstrating its robustness to small changes of vertex positions.

The second experiment is shown in Figure 11, where we remove the left leg surface of the piano and calculate the distance measure based on the original complete model. The distance heatmap shows that our method and oriented bounding box (OBB) are least affected, because our network extracts the global feature through 3D convolutions and pooling, and it learns the global extrinsic shapes of the model. The feature vector of partial piano is close to the complete one because they have very close global shapes. OBB is also insensitive to this situation. The method of Kazhdan *et al.* [4] has some error because its feature is obtained from voxels which are changed. The method of Martinet *et al.* [17] is suitable for accurate symmetry de-

TABLE 3
Comparisons with different methods on partial shape set with large contiguous regions removed. We take the test set and for each shape we randomly choose a radius and a center point, then we remove triangles of the shape that fall inside the sphere. Our method is robust and produces minimum average GTE.

Error	PCA	Oriented Bounding Box [57]	Kazhdan *et.al.* [4]	Martinet *et.al.* [17]	Mitra *et.al.* [29]	PRST [2]	PRST [2] with GEDT	Korman *et.al.* [5]	Our
GTE($\times 10^{-2}$)	9.74	1.43	1.92	16.8	65.1	4.74	5.14	41.7	**0.597**

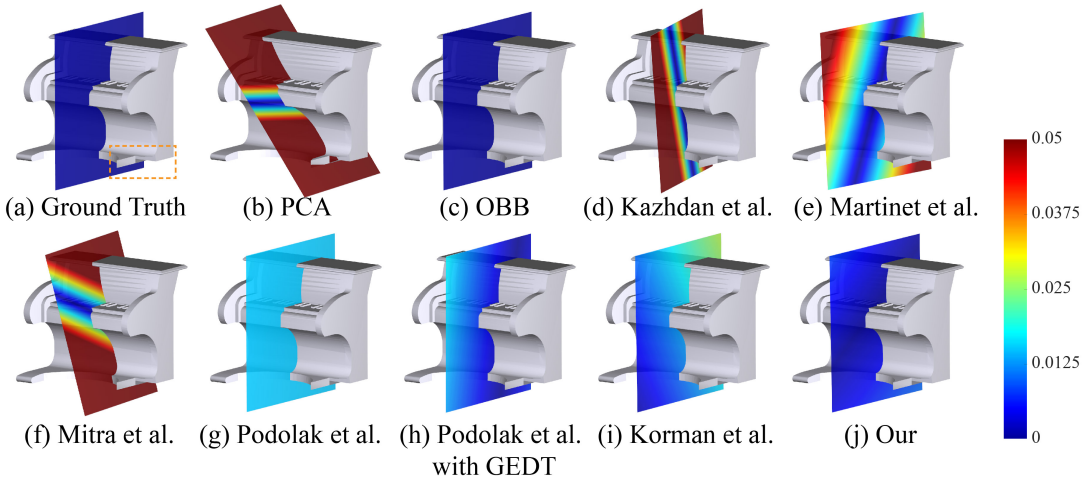


Fig. 11. Robustness testing by removing parts of the model. We remove the left leg of the piano and compare the error based on the complete model. This demonstrates that our network can also produce accurate results even when the input misses large partial models.

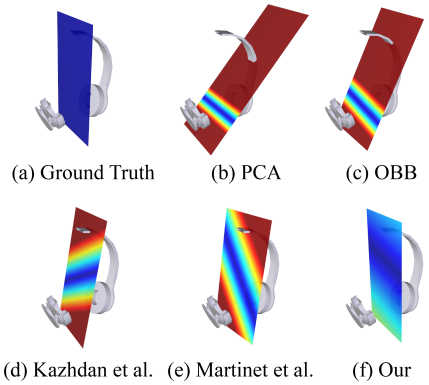


Fig. 12. Robustness testing by removing larger parts of the headphone model. Our network could produce more accurate result because the network is trained on a large number of shapes including headphones and the feature of partial shape is close to other complete headphones, while other methods lack such knowledge, leading to worse results.

tection, and the incomplete shape affects the result significantly. The methods of Podolak *et al.* [2] and Mitra *et al.* [29] use sample points to get the reflection planes, and the distribution of the partial piano points is somewhat different from the complete shape, so the reflection planes have some minor changes. The PCA result is also affected because the shape changes. An example testing robustness against incomplete surfaces is shown in Figure 12. Our network can produce a more accurate result even for the headphone with a substantial part missing, which has the lowest symmetry distance error 1.00×10^{-2}, compared with 2.45×10^{-2} of PCA, 1.53×10^{-2} of OBB, 1.35×10^{-2} of the method of Kazhdan *et al.* [4], and 1.30×10^{-2} of the method of Martinet *et al.* [17].

We further evaluate on a dataset with large continuous regions removed. We take the test set and for each shape we randomly choose a radius and a center point and remove triangles of the shape that fall inside the sphere. The average GTE is reported in Table 3. Our method is robust and produces minimum average GTE. While voxelization is beneficial, the robustness of our method does not come from it alone, as evidenced by our better performance than the method of Podolak *et al.* [2] which also uses voxel data. We show results of symmetry detection for partial shapes in Figure 13, where the blue areas indicate the missing parts. Our method cannot predict the correct symmetry plane of this shape because the overall shape and its voxel representation are significantly changed due to the very large missing part.

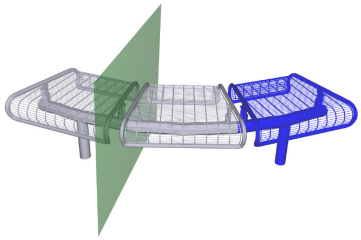


Fig. 13. The visual results on the model with extremely large parts removed. The blue areas indicate the parts removed. Our method fails to predict the correct symmetry plane of the original shape.

6.7 Voxel Resolution and Network Pre-training

The voxel resolution for the CNN can affect the performance of our network. Voxel resolutions from 16^3 to 128^3 have been tested and evaluated with the symmetry distance error on the same test set as in Section 6.1. In this experiment, we

TABLE 4
Comparison of different voxel resolutions. We test the symmetry distance error with resolution from 16^3 to 128^3, and the 32^3 resolution performs best. The error increases with higher resolutions, probably because of overfitting due to limited training data.

Voxel Revolutions	16^3	32^3	64^3	128^3
SDE ($\times 10^{-4}$)	1.16	**0.861**	1.05	1.28

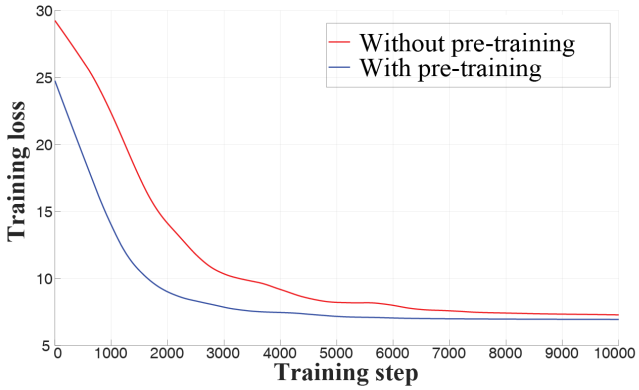


Fig. 14. We compare our network with and without pre-trained weights using the output of the method of Kazhdan *et al.* [4]. We show the training loss (y-axis) w.r.t. training steps (x-axis).

change the number of convolution layers to suit the input voxel size. The number of convolution layers $k_l = log_2(R)$, where R^3 is the input voxel size. As shown in Table 4, the resolution with 32^3 performs best, which is used as the default resolution. The performance drops with higher resolutions, probably due to the overfitting with large number of parameters.

Since the method of Kazhdan *et al.* [4] produces fairly good results, we could use the results to form a supervised loss to train our network and initialize it with these pre-trained weights. As shown in Figure 14, we found that the network with pre-training leads to faster convergence and only requires 5000 steps to converge, compared to the network without the pre-training that needs 9000 steps. Although the accuracy on the test data is nearly identical, the initialization with pre-training is helpful for faster convergence.

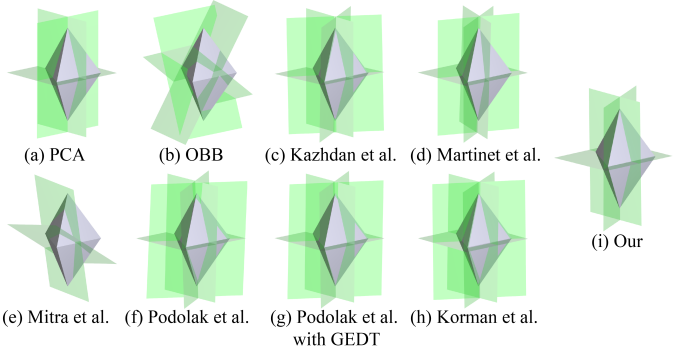


Fig. 15. The detection results of different methods on shapes with more than 3 non-orthogonal symmetry planes. Our method detects three non-orthogonal symmetry planes.

7 Conclusion and Future Work

In this paper, we introduce a novel unsupervised 3D convolutional neural network named PRS-Net, which can discover the planar reflective symmetry of a shape. To achieve this, we develop a symmetry distance loss along with a regularization loss to avoid generating duplicated symmetry planes. We also describe a method to remove invalid and duplicated planes and rotation axes. We demonstrate that our network is robust even when the input has noisy or incomplete surfaces.

Figure 15 shows the detection results of different methods on a shape with more than three, non-orthogonal symmetry planes. Our method detects three non-orthogonal symmetry planes but cannot detect all symmetry planes when the shape has more than three symmetry planes. Although our method could be extended by adding more fully connected layers to predict more symmetry planes, determining the proper number automatically is non-trivial. Our current setting although not perfect is able to handle the majority of cases in practice. In the future, we will investigate *e.g.* reinforcement learning, to predict the number of symmetry planes and their orthogonality, to make the method more general. Moreover, it is interesting to exploit detection of other symmetries including rotational symmetry and intrinsic symmetry using deep learning, which we plan to investigate in the future.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61872440 and No. 61828204), Beijing Municipal Natural Science Foundation (No. L182016), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), Youth Innovation Promotion Association CAS and Tencent AI Lab Rhino-Bird Focused Research Program (No.JR202024).

References

[1] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Symmetry descriptors and 3d shape matching," in *Proceedings of the Eurographics Symposium on Geometry Processing*. ACM, 2004, pp. 115–123.
[2] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A planar-reflective symmetry transform for 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 549–559, 2006.
[3] P. Simari, E. Kalogerakis, and K. Singh, "Folding meshes: Hierarchical mesh segmentation based on planar symmetry," in *Symposium on geometry processing*, vol. 256, 2006, pp. 111–119.
[4] M. Kazhdan, B. Chazelle, D. Dobkin, A. Finkelstein, and T. Funkhouser, "A reflective symmetry descriptor," in *European Conference on Computer Vision*. Springer, 2002, pp. 642–656.
[5] S. Korman, R. Litman, S. Avidan, and A. M. Bronstein, "Probably approximately symmetric: Fast rigid symmetry detection with global guarantees," *Computer Graphics Forum*, 2014.
[6] G. Marola, "On the detection of the axes of symmetry of symmetric and almost symmetric planar images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 1, pp. 104–108, 1989.
[7] C. Sun and D. Si, "Fast reflectional symmetry detection using orientation histograms," *Real-Time Imaging*, vol. 5, no. 1, pp. 63–74, 1999.
[8] G. Loy and J.-O. Eklundh, "Detecting symmetry and symmetric constellations of features," in *European Conference on Computer Vision*. Springer, 2006, pp. 508–521.
[9] J. Liu and Y. Liu, "Curved reflection symmetry detection with self-validation," in *Asian Conference on Computer Vision*. Springer, 2010, pp. 102–114.

[10] S. Lee and Y. Liu, "Curved glide-reflection symmetry detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 2, pp. 266–278, 2011.

[11] D. C. Hauagge and N. Snavely, "Image matching using local symmetry features," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 206–213.

[12] T. Zielke, M. Brauckmann, and W. von Seelen, "Intensity and edge-based symmetry detection applied to car-following," in *European Conference on Computer Vision*. Springer, 1992, pp. 865–873.

[13] K. Fukushima and M. Kikuchi, "Symmetry axis extraction by a neural network," *Neurocomputing*, vol. 69, no. 16-18, pp. 1827–1836, 2006.

[14] C. Funk and Y. Liu, "Beyond planar symmetry: Modeling human perception of reflection and rotation symmetries in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 793–803.

[15] R. K. Vasudevan, O. Dyck, M. Ziatdinov, S. Jesse, N. Laanait, and S. V. Kalinin, "Deep convolutional neural networks for symmetry detection," *Microscopy and Microanalysis*, vol. 24, no. S1, pp. 112–113, 2018.

[16] M. J. Atallah, "On symmetry detection," *IEEE Transactions on Computers*, no. 7, pp. 663–666, 1985.

[17] A. Martinet, C. Soler, N. Holzschuch, and F. X. Sillion, "Accurate detection of symmetries in 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 2, pp. 439–464, 2006.

[18] D. Raviv, A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Full and partial symmetries of non-rigid shapes," *International journal of computer vision*, vol. 89, no. 1, pp. 18–39, 2010.

[19] H. Zabrodsky, S. Peleg, and D. Avnir, "Symmetry as a continuous feature," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1154–1166, 1995.

[20] B. Li, H. Johan, Y. Ye, and Y. Lu, "Efficient 3d reflection symmetry detection: A view-based approach," *Graphical Models*, vol. 83, pp. 2–14, 2016.

[21] A. Ecins, C. Fermuller, and Y. Aloimonos, "Detecting reflectional symmetries in 3d data through symmetrical fitting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1779–1783.

[22] M. Cicconet, D. G. Hildebrand, and H. Elliott, "Finding mirror symmetry via registration and optimal symmetric pairwise assignment of curves: Algorithm and results," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1759–1763.

[23] P. Ji and X. Liu, "A fast and efficient 3D reflection symmetry detector based on neural networks," *Multimedia Tools and Applications*, vol. 78, no. 24, pp. 35 471–35 492, 2019.

[24] M. Ovsjanikov, J. Sun, and L. Guibas, "Global intrinsic symmetries of shapes," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1341–1348, 2008.

[25] V. G. Kim, Y. Lipman, X. Chen, and T. Funkhouser, "Möbius transformations for global intrinsic symmetry analysis," *Computer Graphics Forum*, vol. 29, no. 5, pp. 1689–1700, 2010.

[26] Y. Lipman and T. Funkhouser, "Möbius voting for surface correspondence," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, p. 72, 2009.

[27] N. J. Mitra, A. Bronstein, and M. Bronstein, "Intrinsic regularity detection in 3d geometry," in *European Conference on Computer Vision*. Springer, 2010, pp. 398–410.

[28] R. Gal and D. Cohen-Or, "Salient geometric features for partial shape matching and similarity," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 1, pp. 130–150, 2006.

[29] N. J. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 560–568, 2006.

[30] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas, "Discovering structural regularity in 3d geometry," *ACM transactions on graphics (TOG)*, vol. 27, no. 3, p. 43, 2008.

[31] A. Berner, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel, "A graph-based approach to symmetry detection," in *Volume Graphics*, vol. 40, 2008, pp. 1–8.

[32] Y. Lipman, X. Chen, I. Daubechies, and T. Funkhouser, "Symmetry factored embedding and distance," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 103, 2010.

[33] K. Xu, H. Zhang, A. Tagliasacchi, L. Liu, G. Li, M. Meng, and Y. Xiong, "Partial intrinsic reflectional symmetry of 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 138, 2009.

[34] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 945–953.

[35] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[37] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *European Conference on Computer Vision*. Springer, 2016, pp. 484–499.

[38] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition (CVPR)*, 2016, pp. 5648–5656.

[39] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[40] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[41] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 3, 2017.

[42] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "Planenet: Piece-wise planar reconstruction from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2579–2588.

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[44] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 716–724.

[45] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[46] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3523–3532.

[47] ——, "Prnet: Self-supervised learning for partial-to-partial registration," in *Advances in Neural Information Processing Systems*, 2019, pp. 8812–8824.

[48] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, "Vv-net: Voxel vae net with group convolutions for point cloud segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8500–8508.

[49] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[50] L. Gao, J. Yang, Y.-L. Qiao, Y.-K. Lai, P. L. Rosin, W. Xu, and S. Xia, "Automatic unpaired shape deformation transfer," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 237.

[51] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "Grass: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 52, 2017.

[52] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "Structurenet: hierarchical graph networks for 3d shape generation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–19, 2019.

[53] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang, "SDM-NET: Deep generative network for structured deformable mesh," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–15, 2019.

[54] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, vol. 30, no. 1, 2013, p. 3.

[55] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet:

An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[57] C.-T. Chang, B. Gorissen, and S. Melchior, "Fast oriented bounding box optimization on the rotation group so (3, r)," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 5, p. 122, 2011.
[58] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "ABC: A big CAD model dataset for geometric deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[59] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10,000 3D-printing models," *arXiv preprint arXiv:1605.04797*, 2016.
[60] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM transactions on graphics (TOG)*, vol. 24, no. 3, pp. 408–416, 2005.
[61] J. Huang, H. Su, and L. Guibas, "Robust watertight manifold surface generation method for shapenet models," *arXiv preprint arXiv:1802.01698*, 2018.