Upright-Net: Learning Upright Orientation for 3D Point Cloud

Xufang PANG[1,⋆] Feng LI[3,⋆] Ning DING[1,2,✉] Xiaopin ZHONG[3,✉]

[1]Shenzhen Institute of Artificial Intelligence and Robotics for Society
[2]Institute of Robotics and Intelligent Manufacturing, The Chinese University of Hong Kong
[3]College of Mechatronics and Control Engineering, Shenzhen University

{xufangpang,dingning}@cuhk.edu.cn, 1910294001@email.szu.edu.cn, xzhong@szu.edu.cn

Abstract

A mass of experiments shows that the pose of the input 3D models exerts a tremendous influence on automatic 3D shape analysis. In this paper, we propose Upright-Net, a deep-learning-based approach for estimating the upright orientation of 3D point clouds. Based on a well-known postulate of design states that "form ever follows function", we treat the natural base of an object as a common functional structure, which supports the object in a most commonly seen pose following a set of specific rules, e.g. physical laws, functionality-related geometric properties, semantic cues, and so on. Thus we apply a data-driven deep learning method to automatically encode those rules and formulate the upright orientation estimation problem as a classification model, i.e. extract the points on a 3D model that forms the natural base. And then the upright orientation is computed as the normal of the natural base. Our proposed new approach has three advantages. First, it formulates the continuous orientation estimation task as a discrete classification task while preserving the continuity of the solution space. Second, it automatically learns the comprehensive criteria defining a natural base of general 3D models even with asymmetric geometry. Third, the learned orientation-aware features can serve well in downstream tasks. Results show that our network outperforms previous approaches on orientation estimation and also achieves remarkable generalization capability and transfer capability.

1. Introduction

The upright orientation of an object is associated with its most commonly seen pose in daily life. Such pose usually serves multiple objectives such as functionality, stability, semantic meaning, facility, and so on. Posing object in their upright orientation is the human preference since it makes the objects easily recognizable [12].

In computer graphics and computer vision, registering objects in upright orientation is usually the first step for 3D model analysis [26, 27], which benefits applications like shape matching [1], shape retrieval [19], robotic manipulation on object placement problems [13], generation of thumbnails for 3D shape repositories [11], and so on. With the thriving of deep learning, neural networks operating on point clouds have shown superior performance on these tasks. However, their performance is usually evaluated on a dataset aligned in a canonical frame. A key challenge in learning unaligned point cloud data is to learn features that are invariant or equivariant with respect to geometric transformations [26]. However, either T-Net in PointNet [15] or ITN [26] is not significantly contributive to performance due to their weak supervision on pose transformation. Thus, orientation estimation is used as an auxiliary task for computer graphics tasks, such as shape classification and keypoint prediction [14], to achieve strongly supervised pose transformation.

However, it is an open challenge to recover the upright orientation for general 3D models because of two reasons. Firstly, determining the upright orientation of a 3D object requires a comprehensive consideration of physical laws, geometric properties, semantic preference, functionality, design knowledge, and so on. Thus it is hard to define a universal rule to upright general 3D shapes effectively [12]. For example, paper [5] tried to infer the upright orientation of a 3D model based on a series of hand-crafted geometrical features, however, it failed on some models due to the bias or conflict among features. Secondly, estimating the upright orientation is intuitively a continuous rotation problem, for which, however, the convergence of solutions is fragile due to the diverse variation between different shape categories. Although the divide-and-conquer scheme [12], which first classifies the object and then estimates the orientation via a regression model, achieves an acceptable result, it is not an ideal solution to train separate regression models for different categories.

In this paper, we convert the continuous orientation prob-

⋆ Co-first authors, ✉ Co-corresponding authors

lem into a discrete classification problem based on our observation that objects with mostly commonly seen upright poses are usually supported by their natural bases, where the surface or points contact the supporting plane. Our key idea is to extract points on a 3D model that forms the natural base and then compute its upright orientation as the normal of the fitted plane over the base points, that point to the object. We propose a data-driven deep learning method to automatically extracts a comprehensive feature description that provides sufficient discrimination power for general upright orientation detection. Our method has three advantages. First, formulating the continuous orientation problem as a discrete classification problem improves the generalization ability of the solution. Second, different from hand-crafted features, our deep-learning-based approach automatically encodes comprehensive criteria that define the natural base. Third, by learning orientation-aware features, we can compress the feature space and boost the performance in downstream tasks. Our experiment results indicate that UprightNet outperforms previous approaches on orientation estimation, and demonstrate remarkable generalization ability and transfer capability.

2. Related Works

2.1. Image Orientation

It is an undeniable fact that the orientation of objects on images seriously affects some computer vision tasks, such as image classification [8], object detection [4], face recognition [16], and so on. This is because the orientation changes of objects on images cause large appearance variation, which further boosts the difficulties in constructing feature descriptors.

For years, researchers have devoted themselves to designing rotation-invariant features for building efficient and robust algorithms for classifying or recognizing objects that may appear in images under different orientations [7, 21]. The traditional solution to this problem is to specifically train multiple classifiers at different orientations [7], which is not applicable for wide applications. With the development of deep learning methods, paper [6] proposes a method to generate oriented proposals (OOPs) to reduce the detection error caused by various orientations of the object, which experimentally proves that the orientation registration of objects improves the performance of downstream tasks. Another way to improve model robustness to object orientation variation is overfitting the training dataset, *i.e.* rotating training images with different angles [17]. However, the orientation of objects is a continuous variable, overfitting the training dataset would increase the training burden, especially for the 3D model, which with 3 degrees of freedom.

2.2. Orientation of 3D Models

Many approaches have been proposed for uprighting 3D models, which are generally categorized as three-dimensional geometry processing-based methods and data-driven learning-based methods.

Upright orientation estimation based on geometry processing usually suffers from low robustness due to the conflict or incompletion of hand-crafted features. Principal Component Analysis (PCA) is commonly used for orientation normalization of 3D models, which set the center of mass as the origin and principal axes as the canonical axes. However, it is not robust for general models especially with asymmetric geometry [10] and thus let alone produce compatible alignment with the upright orientation of objects [10]. Based on the observation that the coordinate matrix of a 3D object with upright orientation tends to have reduced rank. Paper [9] proposes a method aligning the 3D shape with axes by iterative rectification of axis-aligned projections as low-rank matrices independently. Paper [22] estimates the upright orientation via minimizing the tensor rank of the 3D shape's voxel representation. However, such rank minimization methods also fail on asymmetric 3D objects as PCA. Trimesh [3], which computes stable orientations of a mesh and their quasi-static probabilities based on the analysis of the center of balance, is also not applicable for our problem, given the lack of consideration of semantic and functional information.

Data-driven learning-based methods are appreciated to deal with general objects. Based on the observation that man-made objects should have a supporting base on which they can be steadily positioned. Both papers [5, 11] extract the candidate bases on 3D models by clustering facets of the simplified convex hull of the 3D model. And then determine the natural base using an assessment function referring to a set of determining factors for base definition, such as geometrical properties, physical laws, and so on. Although their hand-crafted features cover more abundant factors, such kinds of methods cannot deal with general objects, especially for natural objects, this is because it is hard to define a universal rule to upright 3D models via hand-crafted features. With the advent of deep learning, many attempts have been done to explore its effects on orientation estimation. Liu *et al*. [12] apply a regression model to predict the continuous orientation of an object using 3D Convolutional Networks (ConvNets). To conquer the adverse impact of the diverse variation between different shape categories, they first classify the object and then train an individual regression model for each category, which is a heavy burden for real application. Paper [14] approaches the problem from another angle by converting the continuous orientation estimation task into a set of discrete orientation estimation tasks. However, by discretizing the orientation angles, such methods will suffer low discrimination while

using dense sampling, and rough estimation while using sparse sampling. Additionally, upright orientation estimation also gains attention in the robotics manipulation field to guide a machine(agent) to orientate 3D shapes step by step to upright given its current observation [2, 13]. These methods require an iterative process that cooperates with the agent interactions.

In our work, instead of discretizing the orientation angle, we take full advantage of a common functional structure, the natural base of the model, and convert the continuous orientation task into a classification (segmentation) problem to extract the nature base. Based on such a scheme, we simplify the orientation estimation problem but reserve the continuity of the solution space, *i.e*. the upright orientation.

3. Problem Statement

To achieve a continuous solution for upright orientation estimation of general 3D models, we formulate the continuous orientation problem as a delicate classification task, accomplished via two steps. Firstly, it extracts points on 3D models that form the natural base. Secondly, we fit these points with a plane and define the upright orientation of the model with the plane normal directing to the mass center of the model.

Given a point cloud $\boldsymbol{\mathcal{P}} = \{\boldsymbol{p}_i | i = 1, 2, \cdots, n\} \in \mathbb{R}^{n\times 3}$, we first estimate a binary label for each point, and the supporting points on natural base is identified as a subset of the point cloud with positive labels:

$$\boldsymbol{\mathcal{S}} = \{\boldsymbol{p}_i | \hat{y}_i = 1, i = 1, 2, \cdots, m\} \in \mathbb{R}^{m\times 3}, \quad (1)$$

where $\hat{y}_i$ is the output label for point $\boldsymbol{p}_i$ and m denotes the number of predicted supporting points. Then, we fit the supporting points set $\boldsymbol{\mathcal{S}}$ into a natural base via the RANSAC algorithm and then infer the upright orientation of the object. To encode the intricate rules defining the natural base, we design a new network architecture based on EdgeConv [23] and Attention mechanism [20], named as Upright-Net. This architecture directly consumes an unorganized point set as input and predicts the natural base for any general 3D point cloud models.

4. Upright-Net

In this section, we introduce the framework of Upright-Net, including the feature encoding, network architecture, and the loss function.

4.1. Feature Encoding of Point Cloud

Our feature encoding of point cloud should satisfy the following requirements: 1) capturing both geometric feature and semantic information; 2) characterizing regions of interest that benefit the learning tasks. To meet these needs, the Upright-Net first apply classical *EdgeConv Layer*, which learns the local geometric features and groups points in semantic space layer by layer, and then followed by stacked masked Attention Layers for extracting core information and MLP Layer to remap the feature space for better exploration. We introduce these three used modules as follows.

MLP (Multi-layer Perceptron) Layer. We apply MLP for feature embedding, which performs as a 1×1 convolution on each point feature, followed by a batch normalization layer and an activation function:

$$MLP(\cdot) = \sigma\left(BN\left(conv_{1\times 1}(\cdot)\right)\right), \quad (2)$$

where σ denotes the activation function, BN is the batch normalization and $conv$ is the convolution with the subscript indicating the filter size.

EdgeConv Layer. EdgeConv explicitly constructs a local graph and learns the embeddings of local context. For each point feature $\boldsymbol{f}_i$ with d_f dimension, a KNN (K Nearest Neighbors) query of $\boldsymbol{f}_i$ is calculated as $N_k(f_i) = \{f_{i,j} | j = 1, ..., k\} \in \mathbb{R}^{k\times d_f}$, and then a radial locally directed graph is constructed over this subset of points with Edge vectors directed from f_{ij} to f_i. The edge features e_{ij} that combines both global information and local context is computed as:

$$e_{ij} = \left[\boldsymbol{f}_i, \boldsymbol{f}_{ij} - \boldsymbol{f}_i\right] | \forall \boldsymbol{f}_{ij} \in \mathcal{N}_k(\boldsymbol{f}_i), \quad (3)$$

where $[\cdot, \cdot]$ denotes the concatenation of point feature and the edge vector. Given edge features e_{ij}, we first map it into high-dimensional features via a shared MLP Layer, and then apply a max-pooling to aggregate edges features over the local graph. In particular, our EdgeConv layer is implemented as

$$EdgeConv(f_i) = \max_{j\in 1,\ldots,k} \left(MLP(e_{ij})\right). \quad (4)$$

It is noted that f_i represents point features in Euclidean space or in new embedding space.

Attention Layer. Attention mechanism is commonly used for extracting core information in feature map. In Upright-Net, we apply the Scaled Dot-Product Attention [20] to identify the importance of different regions on point cloud. Suppose the layer input is a d_f-dimension feature matrix, denoted as $\boldsymbol{F} = \{\boldsymbol{f}_i | i = 1, 2, \cdots, n\} \in \mathbb{R}^{n\times d_f}$. We first obtain the query matrix $\boldsymbol{Q}$, the key matrix $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$ respectively via three independent *MLP Layers* as follows:

$$\begin{aligned} \boldsymbol{Q} &= \left[MLP_q(\boldsymbol{f_i}) | \forall \boldsymbol{f}_i \in \boldsymbol{F}\right] \in \mathbb{R}^{n\times d_q} \\ \boldsymbol{K} &= \left[MLP_k(\boldsymbol{f_i}) | \forall \boldsymbol{f}_i \in \boldsymbol{F}\right] \in \mathbb{R}^{n\times d_k}, \\ \boldsymbol{V} &= \left[MLP_v(\boldsymbol{f_i}) | \forall \boldsymbol{f}_i \in \boldsymbol{F}\right] \in \mathbb{R}^{n\times d_v} \end{aligned} \quad (5)$$

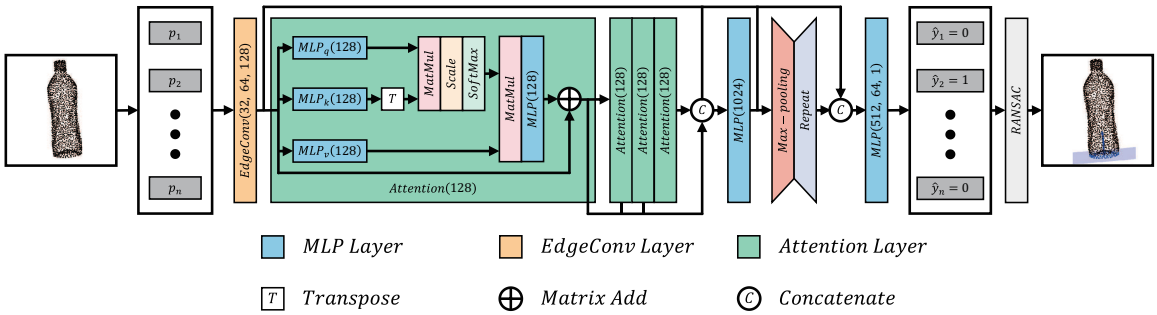


Figure 1. Upright-Net pipeline. The network comprises the *MLP Layers*, the *EdgeConv Layers*, and the *Attention Layers*, and the numbers in the bracket indicate the output channels of the layers. Taking n points as input, the model calculates a segmentation score for each point, indicating whether it is a supporting point or not. After finding all estimated supporting points, we utilize the RANSAC algorithm to fit a natural base and then infer the upright orientation as the normal of the base pointing towards the mass center of a object.

where $[\,\cdot\,]$ denotes a operation for packing vectors together into a matrix. Then, we compute the dot products of $\boldsymbol{Q}$ and the transposed $\boldsymbol{K}$, dividing it by $\sqrt{d_f}$, and apply a softmax function over the result to obtain a weight matrix $\boldsymbol{W}$ indicating the interdependence of pairwise point features.

$$\boldsymbol{W}\left(\boldsymbol{Q},\boldsymbol{K}\right)=softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d_k}}\right)\in\mathbb{R}^{n\times n} \tag{6}$$

With this weight matrix, we compute the weighted feature matrix as $\widetilde{\boldsymbol{F}}=\boldsymbol{W}\cdot\boldsymbol{V}$, where $\widetilde{\boldsymbol{F}}=\left\{\widetilde{\boldsymbol{f}}_i|i=1,2,\cdots,n\right\}\in\mathbb{R}^{n\times d_v}$. By additionally remapping the weighted features via an MLP Layer and using a residual connection to ease the training process, the matrix of output is computed as:

$$Attention\left(\boldsymbol{F}\right)=\left[MLP\left(\widetilde{\boldsymbol{f}}_{\boldsymbol{i}}\right)|\,\forall\widetilde{\boldsymbol{f}}_{\boldsymbol{i}}\in\widetilde{\boldsymbol{F}}\right]+\boldsymbol{F}, \tag{7}$$

where $[\,\cdot\,]$ denotes the operation of packing feature vectors into feature matrix.

4.2. Network Architecture

The Upright-Net takes a point cloud as input and predicts a binary label for each point, indicating whether it is a supporting point. The architecture is shown in Fig. 1, which first applies three EdgeConv Layers to expand the feature dimension and group points in both euclidean and semantic space layer by layer. Then four stacked Attention Layers are used to emphasize core information. By concatenating hierarchical features from different Attention layers, followed by an MLP Layer and a Max-pooling operation, we obtain the global features of the point cloud. After that, the architecture jointly considers the low-level features, weighted features, and global features via liking hierarchical features from different layers to gain a better embedding and prevent the gradient vanishing problem in network training. Then with the predicted supporting points, considering the outlier-robustness property of an estimator, we apply the random sample consensus (RANSAC) algorithm fitting a planar model over the points,

$$\hat{\boldsymbol{\beta}}=RANSAC\left(\boldsymbol{\mathcal{S}}\right). \tag{8}$$

Therefore, the supporting plane, where the natural base is located, is determined by a coefficients vector $\hat{\boldsymbol{\beta}}=(\hat{\boldsymbol{a}},\hat{d})$ with $\hat{\boldsymbol{a}}$ denoting the unit normal of the plane and $\hat{d}$ denoting the distance form the origin to the plane. The upright orientation $\hat{\boldsymbol{o}}=\pm\hat{\boldsymbol{a}}$ is then defined by the direction pointing toward the mass centre of the model. In rare cases, the number of predicted supporting points is less than 3, then the upright orientation is computed as the direction of a vector starting from the mass center of the predicted points pointing to the mass center of the point cloud.

4.3. Loss Function

Our loss function comprise three terms: binary cross-entropy, fitting residual, and stability, i.e.,

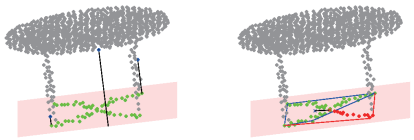
$$\mathcal{L}=\mathcal{L}_{BCE}+\alpha_1\mathcal{L}_{FR}+\alpha_2\mathcal{L}_{Stab}, \tag{9}$$

where α_1 and α_2 are hyper-parameters, seting as $\alpha_1=0.1$ and $\alpha_2=1$ empirically. In this section, we introduce the formulation of each term in details.

Binary Cross-Entropy Loss. The binary cross-entropy term L_{BCE} is used to encourage predicted point label $\hat{y}_i$ to match with the ground truth label y_i:

$$\mathcal{L}_{BCE}=\frac{1}{n}\sum_{i=1}^{n}-\left[y_i\log\left(\hat{y}_i\right)+\left(1-y_i\right)\log\left(1-\hat{y}_i\right)\right], \tag{10}$$

where n is the number of points in point cloud. Comparing to the other two terms, binary cross-entropy loss provide direct and strong feedback in training process and allow the architecture to achieve a fast convergence rate.



(a) Fitting Residual Loss. (b) Stability Loss.

Figure 2. Definition of loss function. The green points indicate the true positives, and the red points denote the false negatives.

Fitting Residual Loss. The point-wise binary cross-entropy loss penalizes wrong predictions equally, no matter it is far or near from the natural base. This is clearly not reasonable in our case since the predictions with a minor displacement may not overturn the result of upright direction estimation but the prediction with large displacement probably can. Therefore, a fitting residual term is used to introduce flexible penalties that encourage positive predictions near the natural base while suppressing the ones that are far away. This term is computed as the average distance between the positive predictions and the ground truth natural base:

$$\mathcal{L}_{FR} = \begin{cases} \frac{1}{m}\sum_{\boldsymbol{p}_i \in \boldsymbol{\mathcal{S}}} \left| \boldsymbol{a}\boldsymbol{p}_i^\top + d \right| & \text{if } m \geq 3 \\ max\_distance & \text{else} \end{cases}, \tag{11}$$

where $\boldsymbol{p}_i$ is a predicted supporting points and $\beta = (\boldsymbol{a}, d)$ is the fitted coefficients vector representation of the ground truth supporting plane. In cases that the number of predicted supporting point is less than 3, the fitting residual loss is then set as a max distance of 2 in our training, which is large than the maximum size of a normalized point cloud.

Stability Loss. It is commonly accepted that the static stability of an object is related to the proportion of its mass located above the natural base, for example, for a roughly symmetric shape, the projection of its mass center on the natural base should be close to the center of the natural base, and for an asymmetry shape, it shows the opposite. However, most standing objects are designed to be reasonably stable against small perturbing forces as long as their center of mass projecting along the vector of upright direction falls on the natural base of the object. Thus to cover rules for general shapes, instead of considering the mass distribution of a shape that is theoretically affected by the materials involved, we simply force the mass distribution of the predicted natural base to be consistent with the mass distribution of the ground truth natural base. Therefore, we define the stability term via a simple geometric estimate:

$$\mathcal{L}_{Stab} = \begin{cases} \|\hat{\boldsymbol{c}} - \boldsymbol{c}\| & \text{if } m \geq 3 \\ max\_distance & \text{else} \end{cases}. \tag{12}$$

where, by projecting the predicted or ground truth supporting points on their fitted plane, $\hat{\boldsymbol{c}}$ or $\boldsymbol{c}$ is computed as the center of the convex hull determined by the projections.

5. Experiment

In this section, we first introduce the dataset applied in our experiments and network implementation details (Sec.5.1). Then we provide detailed experiments to evaluate the performance of our network on upright direction estimation (Sec.5.2). After that, we separately validate the generalization capability (Sec.5.3) and transfer capability (Sec.5.4) of our network. At last, ablation studies are performed to validate our network design (Sec.5.5).

5.1. Dataset and Implementation Details

To train the UprightNet, a large dataset with labeled supporting points is required. Fortunately, we found that most objects in the dataset of ModelNet40 [24] and ShapeNet [25] are placed in an upright standing position in canonical coordinates, i.e. with $+y$ axis pointing to the upright direction. Thus, we first collect a data set by selecting 15 common object categories with an upright standing pose from ModelNet40 [24], where each category keeps 100 shapes, except for Bowl with only 80, see example shapes in Fig. 3. Then with normalized models, we annotate points with coordinate y smaller than a threshold (0.05) as supporting points. To gain a data set covering various possible poses, we rotate each model 100 times by uniform random sampled angles. As a result, we generate a data set with up to 148,000 models (1,480 shapes $\times$ 100 rotations), named as UprightNet15 in the paper with 15 denoting the number of classes. We apportion the data into training and test sets, with a 3-1 split. Additionally, by appliying similar labeling and data augmentation methods, we select 20 shapes in 5 unseen categories from ShapeNet [25] dataset forming a small dataset named UprightNet5 to validate the generalization capability of the network and also use the Original and Rotation sets of RobustPointSet [18] to test the transfer capability of the network.

Upright-Net is implemented in the PyTorch and trained on Nvidia RTX 2080 Ti GPUs. In the training stage, we apply Adam optimizer with a learning rate of 10^{-3} and a batch size of 64. Momentum and weight decay are set to 0.9 and 10^{-4} respectively for 50 epochs.

5.2. Upright Orientation Estimation Performance

We quantitatively evaluate the Upright-Net via two performance measurement metrics. One is Mean Error (ME), which measures the average angular deviation between the estimated upright orientation $\hat{\boldsymbol{o}}_i$ and its ground truth $\boldsymbol{o}_i$:

$$ME = \frac{1}{N}\sum_{i=1}^{N} \arccos\langle \hat{\boldsymbol{o}}_i, \boldsymbol{o}_i \rangle, \tag{13}$$

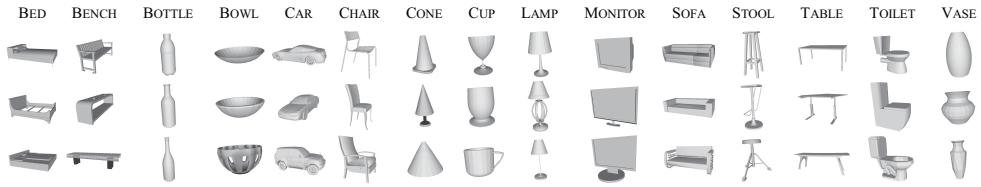


Figure 3. Example shapes of UprightNet15 posing in their upright orientations..

Table 1. The quantitative comparison for upright orientation estimation of our Upright-Net to ConvNets.

	ME (°)	Accuracy (%)
ConvNets [12]	21.52	69.01
Upright-Net	**12.78**	**91.80**

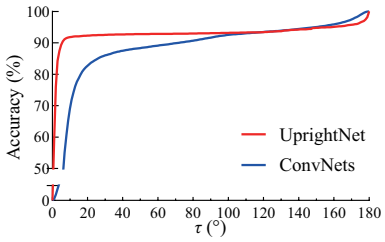


Figure 4. Accuracy visualization with increasing threshold τ.

where $\langle \cdot, \cdot \rangle$ represents the inner product operation, N denotes the number of test models, $\hat{\boldsymbol{o}}_i$ and $\boldsymbol{o}_i$ are unit vectors. We also evaluate the accuracy of upright orientation estimation via computing the proportion of models with angular deviation smaller than a given angle τ:

$$Accuracy = \frac{N_{\arccos \langle \hat{\boldsymbol{o}}_i, \boldsymbol{o}_i \rangle < \tau}}{N}. \tag{14}$$

where τ is set as 10° in our experiments.

In Tab. 1, we compare Upright-Net with ConvNets [12] on upright orientation estimation task. Our network significantly outperforms ConvNets which adopt a divide-and-conquer scheme. There are two possible reasons for this result: first, ConvNets formulate the upright orientation problem as a regression problem, which is hard to train with limited discrete rotational samples even though training different regressor for different categories; second, the erroneous predictions from classification network directly feed the data into a wrong regression network, which probably lead to deviating orientation prediction. In our experiments, the overall classification accuracy of ConvNets achieves 83.89%. On the other hand, Upright-Net applies end-to-end learning and formulate the continuous orientation problem as a classification problem, which is approved an easier training task.

The Upright-Net may perform better than the statistics suggested in Tab. 1. We compare the accuracy of our network and ConvNets via gradually increasing the angular threshold τ and analyzing their error distribution of orientation predictions. As shown in Fig. 4, for Upright-Net, its accuracy is up to 90% at $\tau = 10^\circ$, then remains stable and boosts around $\tau = 180^\circ$, which indicate that the errors are mainly caused by the upside-down predictions, as shown in Fig. 5, and its ME also boosted because of them. On the other side, for regression-based ConvNets, the error predictions are more randomly distributed, which is not perceptually interpretable compared to the results in Fig. 5.

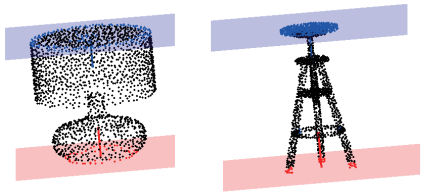


Figure 5. Example models with incorrect orientation predictions. Blue colors the estimated supporting plane, and red colors the ground truth. As we can see that, the upside-down error predictions may be caused by the interfering planar structures on models.

Table 2. Comparison of generalization capability between Upright-Net and ConvNets.

	ConvNets [12]		Upright-Net	
	ME	Accuracy	ME	Accuracy
BASKET	50.16	39.85	23.33	80.55
MICROPHONE	73.73	22.40	20.10	83.15
POT	81.24	18.75	14.55	86.65
SKATEBOARD	65.58	10.75	17.58	63.85
TOWER	37.51	50.25	8.89	92.75
MEAN	61.64	28.40	16.89	81.39

5.3. Generalization Capability

To validate the generalization capability of our network, we test the trained Upright-Net and ConvNets on Upright-Net5, which is introduced in Sec.5.1, consisting of 2000 shapes (20 shapes $\times$ 5 categories $\times$ 100 rotations).

As shown in Tab. 2, the Upright-Net achieves much higher accuracy than ConvNets on all categories. The

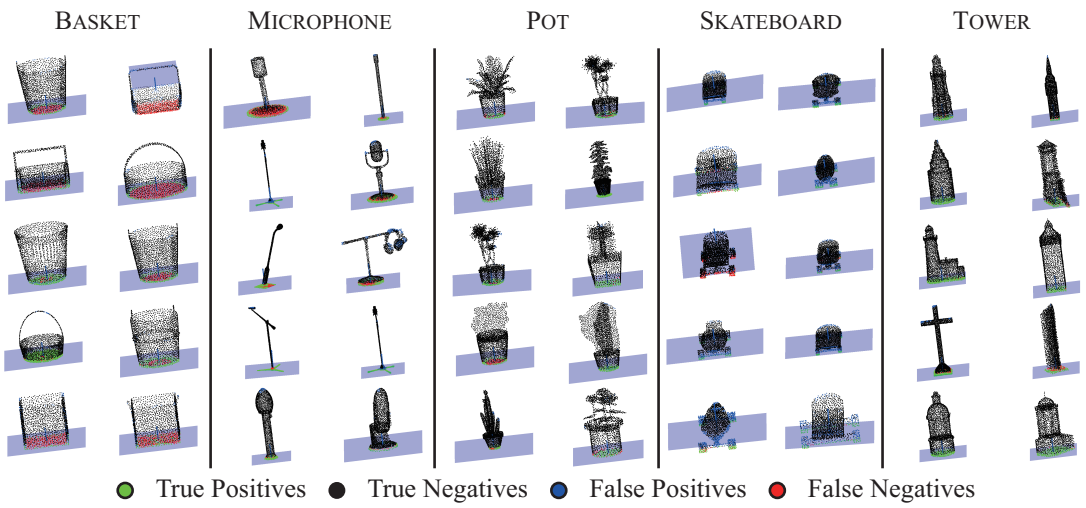


Figure 6. Visualized results of the Upright-Net on unseen object categories in dataset of UprightNet5.

Table 3. Classification results on RobustPointSet.

	Org / Org	Org / Rot	Rot / Rot
PointNet [15]	89.06	8.83	49.03
DGCNN [23]	**92.52**	13.43	58.95
Rotation3D [14]	88.05	8.71	25.57
Upright-Net (fully supervised)	90.94	14.18	67.83
Upright-Net (pre-trained)	87.60	**49.92**	**79.62**

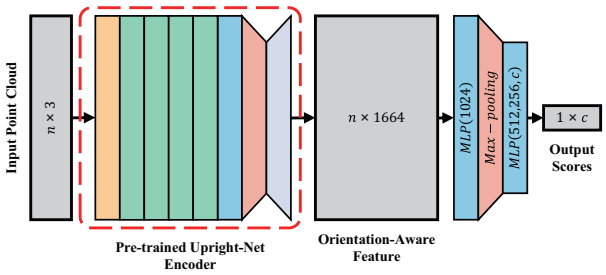


Figure 7. Upright-Net classifier architecture.

Upright-Net treats all categories equally, and it learns both global and local features and emphasis functionality-related geometric structures via constraining the training using specially designed loss terms. Thus for an unseen category, it is the geometrical characteristics of its natural base, not its semantic information that dominates the prediction of Upright-Net. See visualized results of Upright-Net in Fig. 6.

On the other side, the result of ConvNets heavily relies on a category-specific system: a classification network, which classifies BASKET as CUP, MICROPHONE as LAMP, POT as TOILET, SKATEBOARD as BENCH and TOWER as BOTTLE, and category-specific regression networks that apparently overfitted in terms of unseen categories, as a result, its accuracy drops dramatically.

5.4. Transfer Capability

We validate the transfer capability of Upright-Net via a downstream object classification task on the dataset of RobustPointSet [18]. As shown in Fig. 7, we use the pre-trained Upright-Net encoder for feature embedding and then fine-tune the *MLP Layers* to gain classification scores for c categories. We compare our approach to fully supervised methods [15,23] and the self-supervised method [14]. Since the previous methods are usually evaluated on the original dataset or dataset with rotation augmentation in a small angle range, for the sake of fairness, we perform the comparison under three different settings: 1) train and test models both on original RobustPointSet, denoted as Org/Org; 2) train models on the original dataset, but test them on the rotationally augmented dataset, denoted as Org/Rot; 3) train and test models both on the rotationally augmented dataset, denoted as Rot/Rot. We rotate each shape of RobustPointSet 100 times by uniform random sampled angles to generate rotationally augmented dataset.

Tab. 3 shows the results of the experiments. Upright-Net obtains comparable accuracy on the original RobustPointSet (Org/Org) while achieving enormous superiority on the rotational augmented dataset (Rot/Rot). It suggests that our network gains much stronger expressive power on rotational variations than other designs. Moreover, with a pre-trained encoder, the performance of Upright-Net is boosted and significantly outperforms other approaches on the rotational augmented dataset (Org/Rot, Rot/Rot). Therefore, we can conclude that our network is capable

Table 4. Results of upright orientation estimation using different segmentation backbones.

	ME (°)	Accuracy (%)
PointNet	24.80	77.68
DGCNN	13.24	90.26
Upright-Net	**12.78**	**91.80**

Table 5. Hyperparameters tuning of Upright-Net.

α_1	α_2	ME (°)	Accuracy (%)
0	0	20.98	85.13
	0.1	16.43	88.87
	1	15.59	88.37
0.1	0	14.91	89.47
	0.1	12.81	91.74
	1	12.78	91.80
1	0	14.95	89.01
	0.1	**11.87**	**92.47**
	1	13.57	91.22

of learning orientation-aware features of 3D models, which boosts its transfer capability in downstream tasks.

5.5. Ablation Study

In this subsection, we first verify the effectiveness of our Upright-Net backbone, then we analyze the hyper-parameter of the loss function, and visualize the feature map learned inside Upright-Net.

Comparison of segmentation backbones. We compare Upright-Net with PointNet [15] and the DGCNN [23] for supporting points segmentation. Our experiment results suggest that the Upright-Net achieves the lowest ME (Mean Error) and highest accuracy, see Tab. 4. We believe it benefits from its *EdgeConv Layers*, which extract local features, and the *Attention mechanism* that emphasizes functional-related geometrical features. Besides, linking hierarchical features from different layers allow the network to consider all the global, local, and weighted features to gain a more informative embedding.

Hyperparameters tuning. The hyperparameters α_1 and α_2 in the loss function are tuned by running the whole training job with different parameter settings and selecting the best performance setup. Tab. 5 indicate that Upright-Net trained with $\alpha_1 = 1$ and $\alpha_2 = 0.1$ deliver the best performance on test set, however, we found it performed poorer on unseen categories. Therefore, we set $\alpha_1 = 0.1$ and $\alpha_2 = 1$ in our model for a reasonable trade-off between estimation performance and generalization ability.

Visualization. We visualize the concatenated features before the last MLP layer of Upright-Net with a heatmap representing feature averages. As shown in Fig. 8, the points close to the natural base gain higher responses as we expected, and the points closing to the upright axis also gain high, which is an explainable and positive phenomenon, as it represents the strong response to the stability term.

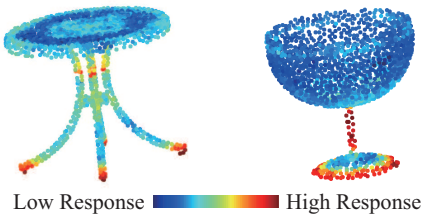


Figure 8. Visualization of the feature map learned by Upright-Net.

6. Conclusion and Limitation

In this paper, we formulate the continuous upright orientation problem as a classification model while still preserving the continuity of the solution space. To encode the intricate rules defining the natural base of objects, we propose a new simple network architecture, the Upright-Net, based on EdgeConvs and Attention mechanism, and constrained by a special design loss function. The experiments indicate that our network outperforms previous approaches on both orientation estimation and downstream classification tasks. Besides, we can easily extend our framework into a multi-class setting in the case of objects with multiple natural bases. However, in rare cases of upside-down predictions may suggest that our model did not gain enough semantic cues for instances with interfering planar structures, thus further study on balancing semantic and geometrical features via improvements on three aspects is valuable: balancing training data, architecture capturing more semantics, loss providing much stronger cues. Another limitation is that our dataset is composed of complete shapes, further experiments on partial point cloud would be interesting and meaningful. To encourage future works, we release our dataset on our website[⊛].

7. Acknowledgements

We thank all the anonymous reviewers and Area Chairs for their valuable comments. This work was supported by the National Key R&D Program of China (grant 2021YFE0204200), the National Natural Science Foundation of China (grant 62106155, U1813216, 61806190), Guangdong Basic and Applied Basic Research Foundation (grant 2021B1515420005, 2021A1515010926), Shenzhen Science and Technology Program (grant JCYJ20180305123922293, JSGG20210802152801004).

[⊛] https://airs.cuhk.edu.cn/en/article/798

References

[1] Silvia Biasotti, Simone Marini, Michela Mortara, Giuseppe Patane, Michela Spagnuolo, and Bianca Falcidieno. 3d shape matching through topological structures. In *International conference on discrete geometry for computer imagery*, pages 194–203. Springer, 2003. 1

[2] Luanmin Chen, Juzhan Xu, Chuan Wang, Haibin Huang, Hui Huang, and Ruizhen Hu. UprightRL: Upright Orientation Estimation of 3D Shapes via Reinforcement Learning. *Computer Graphics Forum*, 40(7):265–275, 2021. 3

[3] Michael Dawson-Haggerty. MS Windows NT michael dawson-haggerty. https://trimsh.org/trimesh.poses.html, 2020. Accessed: 2022-01-28. 2

[4] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2019. 2

[5] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. Upright orientation of man-made objects. In *ACM SIGGRAPH 2008 papers*, pages 1–7. 2008. 1, 2

[6] Shengfeng He and Rynson WH Lau. Oriented object proposals. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 280–288, 2015. 2

[7] Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. Vector boosting for rotation invariant multi-view face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 446–453. IEEE, 2005. 2

[8] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 2

[9] Yong Jin, Qingbiao Wu, and Ligang Liu. Unsupervised upright orientation of man-made models. *Graphical Models*, 74(4):99–108, 2012. 2

[10] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003. 2

[11] Cong-Kai Lin and Wen-Kai Tai. Automatic upright orientation and good view recognition for 3d man-made models. *Pattern recognition*, 45(4):1524–1530, 2012. 1, 2

[12] Zishun Liu, Juyong Zhang, and Ligang Liu. Upright orientation of 3d shapes with convolutional networks. *Graphical Models*, 85:22–29, 2016. 1, 2, 6

[13] Rhys Newbury, Kerry He, Akansel Cosgun, and Tom Drummond. Learning to place objects onto flat surfaces in upright orientations. *IEEE Robotics and Automation Letters*, 6(3):4377–4384, 2021. 1, 3

[14] Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. In *2020 International Conference on 3D Vision (3DV)*, pages 1018–1028. IEEE, 2020. 1, 2, 7

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017. 1, 7, 8

[16] Yichun Shi, Xiang Yu, Kihyuk Sohn, Manmohan Chandraker, and Anil K. Jain. Towards universal representation learning for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[17] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. 2

[18] Saeid Asgari Taghanaki, Jieliang Luo, Ran Zhang, Ye Wang, Pradeep Kumar Jayaraman, and Krishna Murthy Jatavallabhula. Robustpointset: A dataset for benchmarking robustness of point cloud classifiers. *arXiv preprint arXiv:2011.11572*, 2020. 5, 7

[19] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008. 1

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[21] Michael Villamizar, Francesc Moreno-Noguer, Juan Andrade-Cetto, and Alberto Sanfeliu. Efficient rotation invariant object detection using boosted random ferns. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1038–1045. IEEE, 2010. 2

[22] Weiming Wang, Xiuping Liu, and Ligang Liu. Upright orientation of 3d shapes via tensor rank minimization. *Journal of Mechanical Science and Technology*, 28(7):2469–2477, 2014. 2

[23] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 3, 7, 8

[24] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 5

[25] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 5

[26] Wentao Yuan, David Held, Christoph Mertz, and Martial Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018. 1

[27] Keyang Zhou, Bharat Lal Bhatnagar, Bernt Schiele, and Gerard Pons-Moll. Adjoint rigid transform network: Task-conditioned alignment of 3d shapes. *arXiv preprint arXiv:2102.01161*, 2021. 1