Graphical Models 85 (2016) 22–29

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

# Upright orientation of 3D shapes with Convolutional Networks

Zishun Liu*, Juyong Zhang, Ligang Liu

*University of Science and Technology of China, Hefei, Anhui, China*

ARTICLE INFO

*Article history:*
Received 30 November 2015
Revised 1 March 2016
Accepted 6 March 2016
Available online 14 March 2016

*Keywords:*
Upright orientation
Data-driven shape analysis
Voxelization
Convolutional Networks

ABSTRACT

Posing objects in their upright orientations is the very first step of 3D shape analysis. However, 3D models in existing repositories may be far from their right orientations due to various reasons. In this paper, we present a data-driven method for 3D object upright orientation estimation using 3D Convolutional Networks (ConvNets), and the method is designed in the style of *divide-and-conquer* due to the *interference effect*. Thanks to the public big 3D datasets and the feature learning ability of ConvNets, our method can handle not only man-made objects but also natural ones. Besides, without any regularity assumptions, our method can deal with asymmetric and several other failure cases of existing approaches. Furthermore, a distance based clustering technique is proposed to reduce the memory cost and a test-time augmentation procedure is used to improve the accuracy. Its efficiency and effectiveness are demonstrated in the experimental results.



## 1. Introduction

Most objects are usually posed in their upright orientations, which makes them easily recognizable. Also, it is the very first step to pose the given 3D shapes in their upright orientations (Fig. 1) in many graphics and robotics tasks, such as matching [2], retrieval [13,28], shape analysis [34] and placement planning [17]. Moreover, it can be used to generate recognizable object thumbnails, helping the management of 3D shape repositories. Due to various reasons such as modeling platforms or scanning systems, many models in existing databases are not in their upright orientation. Therefore, a number of approaches have been proposed to handle this problem. However, these methods are usually limited to shapes with some regularity and take several seconds to process each shape. Thus more efficient and effective methods are needed.

In this paper, we present a learning based method to predict the upright orientation using 3D Convolutional Networks (ConvNets). Given voxel representations of 3D shapes and corresponding orientation vectors, this prediction task can be formulated as a regression problem. Leveraging the learning ability of deep neural networks, general categories of 3D shapes can be handled without making any assumptions such as symmetry or parallelism. Besides mesh models, the proposed method can deal with shapes represented in other types that can be voxelized, such as implicit surfaces and point clouds, without surface reconstruction [7].

Compared with the ConvNets based approach, existing methods are limited by their predefined rules. For example, the method proposed by Fu et al. [8] is based on the observation that man-made object should have a supporting base on which it can be steadily positioned. Nevertheless, this observation is not applicable to all shapes, especially natural ones. Thus learning based methods are appreciated to deal with general objects. Although the idea of data-driven is adopted in Fu et al. [8], the learning procedure is based on the hand-crafted features such as

* Corresponding author.
*E-mail address:* zishun@mail.ustc.edu.cn (Z. Liu).

http://dx.doi.org/10.1016/j.gmod.2016.03.001
1524-0703/

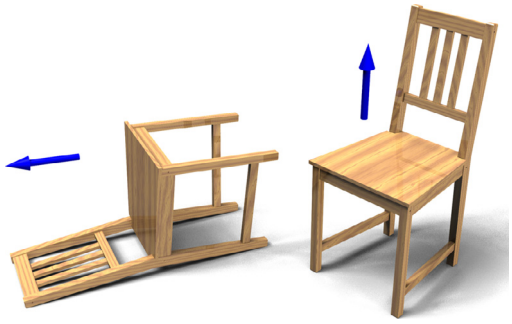

**Fig. 1.** Upright orientation estimation.

stability, visibility and parallelism, which fall into the field of feature engineering. In one word, it is hard to define a universal rule to upright general 3D shapes effectively. By contrast, neural networks work in the style of end-to-end learning. High-level knowledges can be captured from raw data, without relying on object's regularity such as explicit symmetry.

However, a single ConvNet does not work well for all types of shapes. The key challenge is that each shape category exhibits particular characteristic on the upright orientation, for example, cars tend to be horizontal while bicycles are likely to be vertical. This is referred to as *interference effect* [14] which will lead to poor generalization. In other words, different strategies should be taken to handle diverse categories. Thus a *divide-and-conquer* scheme is used in our system. Each shape is first classified by a network and then fed into one of the orientation regression networks that are trained on each of the categories. Furthermore, a distance based clustering method is proposed to reduce the number of networks and a novel test-time augmentation procedure is used to improve the accuracy.

The efficiency and effectiveness of this approach are demonstrated by extensive experiments. Our system achieved the accuracy of more than 90% on the test data and showed the generalization capability of inferring upright orientations for shapes not belonging to the training categories. Also experimental results showed that our system is able to handle several cases that other methods fail. Moreover, estimation for each shape took no more than 0.15 s on average, which is much faster than existing approaches, thus applicable to robotics tasks in which immediate feedback is required.

The main contributions of our approach are summarized in the following.

- General objects can be handled by this approach thanks to the learning ability of ConvNets, including asymmetric shapes.
- The proposed method is at least 30 times faster than existing methods.

The remainder of this paper is structured as follows. Section 2 briefly reviews several related works. In Section 3 our network system is specified. The experimental results and comparisons with related works are demonstrated in Section 4. Finally, Section 5 presents our conclusions and directions of future work to improve our method.

## 2. Related work

**Orientation of images.** Images may differ from their correct orientations by 0°, 90°, 180°, or 270° [3,23,24,32]. Therefore, the image orientation detection problem can be formulated as a four-class classification problem. Most of the existing approaches extract high dimensional feature vector in each possible orientation and then train support vector machines (SVM) [23,32] or other classifiers [3] on feature vectors to detect correct orientation. However, it is difficult to reduce the two-dimensional orientation space to a few candidates for general 3D objects. Thus we formulate the upright orientation estimation of 3D models as a regression problem.

**Upright orientation of 3D models.** In computer graphics, several methods have been proposed to estimate upright orientation or align the given models. One commonly used method is the principal component analysis (PCA) [19] which is inaccurate and not robust for many models, especially asymmetric ones. In Fu et al. [8] and Lin and Tai [22], upright orientation is estimated using supporting base candidates on which a 3D model can stand upright. These methods work well for most of the man-made models while not applicable to natural objects whose supporting bases are not well defined. Another type of method is based on the observation that the coordinate matrix of the 3D object with upright orientation should have reduced rank. Inspired by [37], Jin et al. [18] present an algorithm in which a 3D shape is aligned with axes by iterative rectification of axis-aligned projections as low-rank matrices independently. In Wang et al. [31], a method is proposed by minimizing the tensor rank of the 3D shape's voxel representation. Both methods can handle shapes that have some kinds of symmetries. We can see that none of the above methods is able to deal with general objects.

**Viewpoint selection.** Representative viewpoint provides the most informative and intuitive view of a 3D shape, which benefits many geometry processing applications like shape retrieval. Most approaches select representative viewpoints using geometric information of the 3D models, such as number of visible polygons [25] and silhouette contours [1]. Some works are based on information theory, such as viewpoint entropy [29], multi-scale entropy [30], and viewpoint mutual information [6]. It will be much easier to select the representative views for 3D models if they are posed at the upright orientation by our method.

**3D shape matching, retrieval and registration.** 3D shape retrieval [13,28] and matching [2] techniques attempt to find the similar shapes from databases with queries. 3D shape registration techniques [36] make efforts to find corresponding parts of multiple models. These methods are trying to design a robust and efficient method for measuring the similarity between two shapes or parts over the space of all transformations [19]. To address this issue, most techniques pre-align the models into a common coordinate frame, typically using PCA alignment.

Since our orientation estimation approach predicts a consistent upright orientation for models, it is able to reduce the orientation alignment problem from two to one degree of freedom.

**Deep neural networks.** For computer vision tasks, deep neural networks, especially convolutional networks, have demonstrated excellent performance, by taking 2D images (RGB or RGBD) as input [9,20]. It is only very recent that a few works attempt to tackle 3D shapes related problems via deep learning methods, such as classification, recognition and retrieval. However, most of the works treat 3D shape as a series of multi-view color/depth images [5,27,38], discarding the 3D relationship between different frames. To the best of our knowledge, Wu et al. [33] is the first paper that take volumetric data as input of neural networks, which propose to represent a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid, using a Convolutional Deep Belief Network (CDBN), obtaining good results on shape classification. Another type of 3D Convolutional Network is proposed by [15] for human action recognition in videos, treating time as the third dimension.

## 3. Approach

### 3.1. System overview

Taking $n$ classes of 3D shapes $\mathcal{C}_i$ $(i = 1, \ldots, n)$ into account, the problem of upright orientation estimation is formulated as a regression task. Given a quantity of voxel representations $\mathbf{V}$ of 3D shapes and corresponding unit vectors $\mathbf{u}$ of upright orientation, a function $\mathbf{u} \approx f_{\boldsymbol{\beta}}(\mathbf{V})$ with unknown parameters $\boldsymbol{\beta}$ should be estimated to fit the data.

3D ConvNets can be straightforwardly applied onto this problem. However, due to different shape categories exhibit particular characteristics on their upright orientations, strong interference effects occur that lead to poor generalization [14]. It is difficult to train a universal network which works well for all the $n$ shape categories. Therefore, this task should be accomplished in the style of *divide-and-conquer*, namely, training different networks on different shape categories. Naturally, $n$ regression networks can be trained separately. Moreover, a classification network should be trained to work as a gate by predicting which regression network should be applied onto the input shape. Fig. 2 shows the test stage of the system.

### 3.2. 3D Convolutional Networks

We use the standard architecture of ConvNets for regression and classification.

The regression network takes voxel representations of 3D shapes as input, and the 3D vectors of predicted upright orientation as the output. As illustrated in Fig. 3, the regression network consists of a number of 3D convolution layers and fully-connected layers, each of them is followed by a layer of activation units. The hyperbolic tangent $\tanh(\cdot)$ is chosen as the activation function in the output layer. To avoid slow learning when output values are close to 1 and $-1$, the orientation vectors $\mathbf{u}$ are rescaled by 0.5. In

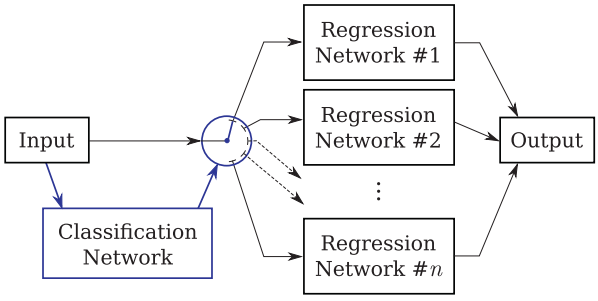


**Fig. 2.** System overview. Input data is propagated through the classification network (blue part) and a class label prediction $i \in \{1, \ldots, n\}$ is obtained. After that, the input data is fed into the $i$th regression networks. The 3D output vector of the regression network is the predicted upright orientation. Each of the $n + 1$ networks can be trained independently. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

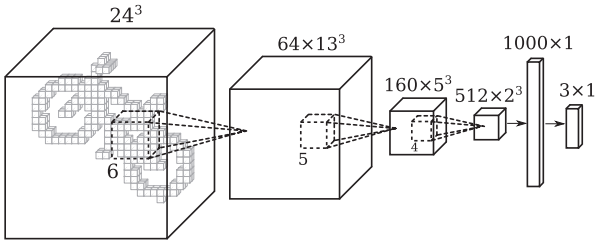


**Fig. 3.** Architecture of the regression network.

the other layers, we choose rectifier [21]

$$\mathrm{ReLU}(x) = \max(0, x)$$

as the activation function. In the end, this network is trained to minimize the Euclidean loss function, using mini-batch gradient descent with batch size $N$.

$$\mathrm{Loss} = \frac{1}{2N} \sum_{i=1}^{N} \|\hat{\mathbf{u}}_i - \mathbf{u}_i\|_2^2,$$

where $\hat{\mathbf{u}}_i$ is the ground-truth three dimensional orientation vector and $\mathbf{u}_i$ is the corresponding regression value.

The classification network shares a similar architecture with the regression networks. The differences are that the output of the last fully-connected layer is fed to a $n$-way softmax which produces a distribution over the $n$ class labels, and a multinomial logistic loss layer is used.

### 3.3. Clustering of shape categories

Although different shape categories exhibit particular characteristics, some categories, such as chair and table, may be handled by similar strategy to find their upright orientations. Those categories can be clustered together and processed with the same regression network. As a result, redundant networks can be removed and then the memory cost can be reduced.

However, it is nontrivial to determine which categories are consistent and which are not. If a network is trained on inconsistent categories, the sacrifice of accuracy would be dramatic compared to the networks trained on each category separately. We propose a clustering strategy based on

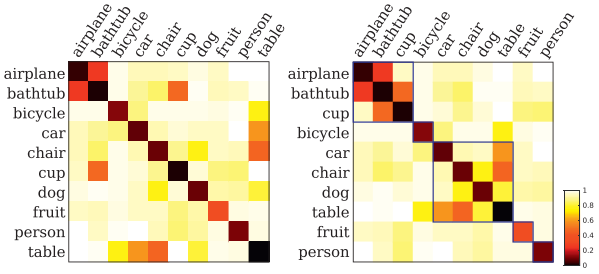


**Fig. 4.** Shape category clustering based on the distance measure defined by error rate. Left: original distance matrix. Right: agglomerated categories are collected in blue squares after clustering. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the distance measure defined by the error rate of each regression network on the other shape categories.

To define the distance measure, we first evaluate the $n$ regression networks on all of the $n$ categories to get a square matrix $\mathbf{E}$ in which $\mathbf{E}(i, j)$ is the error (i.e., $\angle(\mathbf{u}, \hat{\mathbf{u}})$ is larger than some threshold) rate of regression network $\mathcal{R}_i$ on shape category $\mathcal{C}_j$. Then we get the symmetric matrix $\mathbf{D} = (\mathbf{E} + \mathbf{E}^T)/2$ in which $\mathbf{D}(i, j)$ measures the distance between shape categories $\mathcal{C}_i$ and $\mathcal{C}_j$. The shorter between two object categories, the more likely their upright orientations can be estimated with the same network.

Once we obtain the distance matrix, a hierarchical agglomerative clustering algorithm [11] is performed, after which a cluster tree is constructed. Then, we should determine where to cut the hierarchical tree into a number of clusters. At last, new regression networks should be trained on agglomerated shape category clusters while those nets for categories left in their own cluster can be kept. Also, the classification network do not need to be retrained.

### 3.4. Test-time augmentation

For classification tasks, test-time augmentation (TTA) has been proposed to improve the accuracy by taking average of the output from many virtual samples in [4]. From the points of prediction error in Fig. 5, we find some outliers from the results of shape's different input poses. Therefore it would be helpful to take test-time augmentation and average the results in someway robust to outliers, such as taking their median, i.e., 1-norm average.

Given a test shape $\mathcal{S}$, we augment it by transforming with randomly generated rotation matrices $\mathbf{R}_i$ $(i = 1, \ldots, m)$. Then $m$ correspondence voxel representations $\mathbf{V}_i$ are fed into the network system. We classify them into the same class by majority voting of $m$ predicted labels and put them into the same regression network. After getting $m$ regression predictions $\tilde{\mathbf{u}}_i$, we map them back into the coordinate frame of $\mathcal{S}$ as $\mathbf{u}_i = \mathbf{R}_i^{-1}\tilde{\mathbf{u}}_i$. By minimizing the objective function defined below,

$$\mathbf{u}^* = \text{argmin}_{\|\mathbf{u}\|=1} \sum_{i=1}^{m} \angle(\mathbf{u}, \mathbf{u}_i),$$

a better prediction $\mathbf{u}^*$ is expected to be obtained. In existing works, Weiszfeld algorithm [10] was proposed to solve this optimization problem in an iterative manner. However, we replace it with the following reduced version, which is much simpler to solve and works well.

$$\mathbf{u}^* = \text{argmin}_{\mathbf{u}_j,\, j=1,\ldots,m} \sum_{i=1}^{m} \angle(\mathbf{u}_j, \mathbf{u}_i).$$

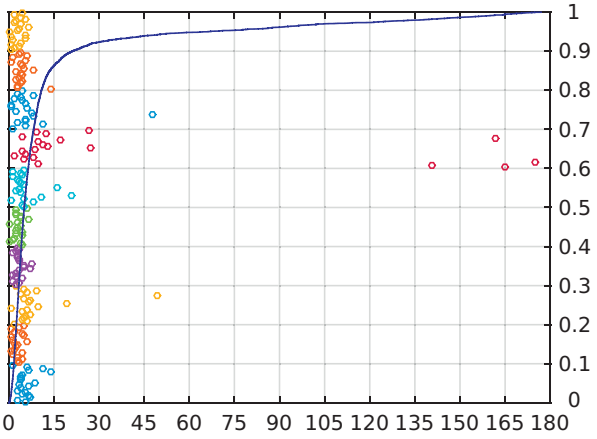


**Fig. 5.** Angle error (in degrees) distribution when using 10 regression networks. Blue curve shows the cumulative distribution function of errors. Small circles represent the error of different shapes with varying poses. We sampled 10 shapes from the test set. Circles in the same color represent the results of the same shape's different poses. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

## 4. Experiments

### 4.1. Implementation

We chose 10 common object categories with unambiguous upright orientation from Princeton ModelNet [33]. Each category contained 100 shapes and was split into training set and test set randomly. The training shapes were rotated 100 times for data augmentation. The test data was also rotated 20 times to study the robustness of this approach to the pose of input shape. Fig. 6 displays some objects sampled from our test set. All experimental results presented in this paper have been tested on a desktop with an Intel(R) Core(TM) i5-4570 CPU @ 3.20 GHz, 8 GB RAM and an NVIDIA GeForce GTX 760 GPU.

A 3D shape is represented as a 24 × 24 × 24 voxel grid. The architecture of our regression network is briefly illustrated in Fig. 3. First we place three convolutional layers, and each of them is followed by a layer of rectified linear units (ReLU). Then two more fully-connected layers are appended. Dropout [26] is applied on the first fully-connected layer. The last layer has 3 output units corresponding to the 3-dimensional orientation label. Such a networks contains 10.6 million floating point parameters, costing 42.6 MB memory. The classification network shares a similar architecture. The details of the designed networks are listed in Tables 1 and 2. We select the network architectures experimentally. Results in different architectures are presented in the supplementary material.

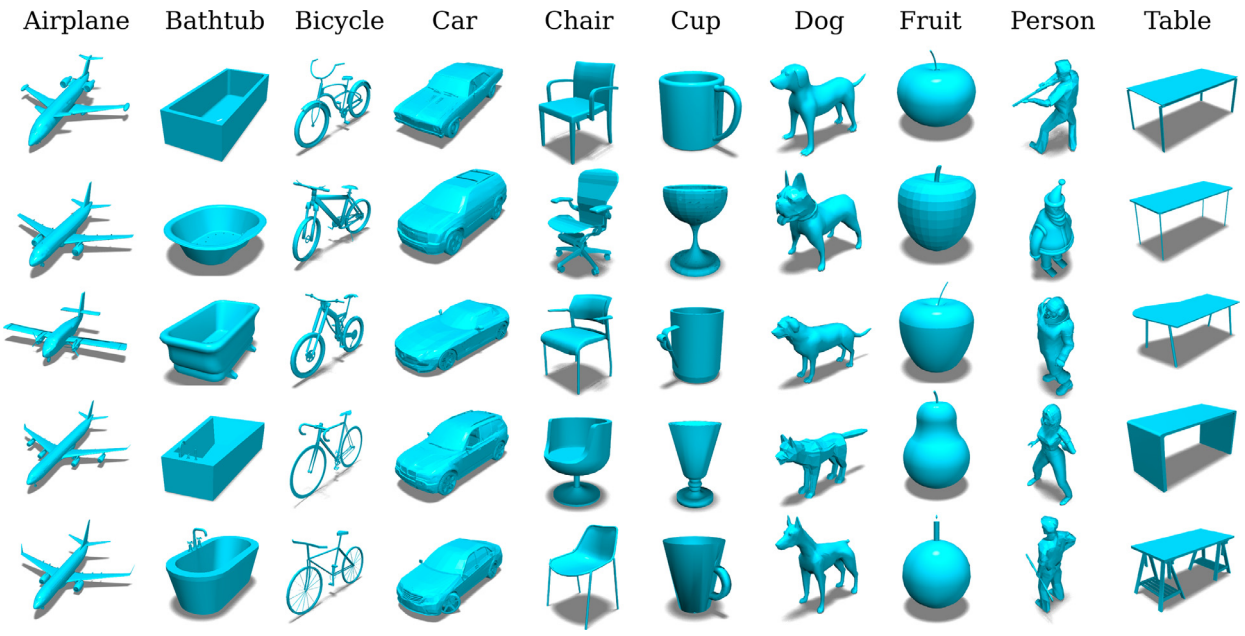


**Fig. 6.** Example shapes from the test set posed in their correct upright orientations found by our method (using five regression networks, without test-time augmentation).

**Table 1**
Architecture of regression network. FC is for fully-connected layer.

| Ind | Type | Filter size | Num | Stride | Pad |
|---|---|---|---|---|---|
| 1 | Conv | $6 \times 6 \times 6$ | 64 | 2 | 3 |
| 2 | ReLU | – | – | – | – |
| 3 | Conv | $5 \times 5 \times 5$ | 160 | 2 | 0 |
| 4 | ReLU | – | – | – | – |
| 5 | Conv | $4 \times 4 \times 4$ | 512 | 1 | 0 |
| 6 | ReLU | – | – | – | – |
| 7 | FC | – | 1000 | – | – |
| 8 | ReLU | – | – | – | – |
| 9 | Dropout | (rate 0.5) | – | – | – |
| 10 | FC | – | 3 | – | – |
| 11 | TanH | – | – | – | – |

**Table 2**
Architecture of classification network.

| Ind | Type | Filter Size | Num | Stride | Pad |
|---|---|---|---|---|---|
| 1 | Conv | $6 \times 6 \times 6$ | 64 | 2 | 3 |
| 2 | ReLU | – | – | – | – |
| 3 | Conv | $5 \times 5 \times 5$ | 256 | 2 | 0 |
| 4 | ReLU | – | – | – | – |
| 5 | FC | – | 512 | – | – |
| 6 | ReLU | – | – | – | – |
| 7 | Dropout | (rate 0.5) | – | – | – |

The networks were implemented with the deep learning framework Caffe [16].

After training the networks for classification and regression, the distance matrix on the shape categories was computed. Based on the distance measurement, we performed the agglomerative clustering algorithm. As a result, 10 shape categories were partitioned into five clusters. (We cut the cluster tree into five clusters empirically. Intuitively, the fewer clusters are left, the stronger interference effect would arise.) Then new regression networks were trained with the same architecture and half of the parameters for regression were saved. The four-legged/wheeled object categories (i.e., car, chair, dog and table) were collected into the same cluster, while the cup-shaped shape categories (i.e., bathtub and cup) were collected into another cluster (this cluster also contains airplane). As each regression network costs 42.6 MB memory and five networks were used instead of 10, about 213 MB memory was reduced. The threshold used for distance measurement was 15°, which should be enough for most graphics and robotics tasks. The distance matrix and result of clustering are shown in Fig. 4.

The classification network was trained eight epochs and achieved the accuracy of 95.6%. Each regression network was trained around 30 epochs. The final accuracy tested on each category of the entire system, which was combined with the classification network and the regression networks, is listed in Table 3. Error distribution in degrees is presented in Fig. 5. The results of test-time augmentation are also shown in Table 3. We rotated each input shape 10 times and the accuracy was improved about 6%. Moreover, TTA would help to give a reasonable result if the regression network's output of some pose degenerates, i.e., producing zero vector (although it had never arisen through our experiments).

### 4.2. Performance analysis

**Interference effects.** To demonstrate the effect of interference, we compared the training of regression networks on two groups of shape categories using the same architecture as before (Table 1). The first group (group A) contains

**Table 3**
Accuracy of the entire system in a variety of settings. The test cases in which $\angle(\mathbf{u}, \hat{\mathbf{u}}) < 15^\circ$ are accounted as correct. Nets-10 is for the system with 10 regression networks for every shape categories. Nets-5 is for the system with five regression networks for every category clusters. TTA is for systems with test-time augmentation.

| | Airplane | Bathtub | Bicycle | Car | Chair | Cup | Dog | Fruit | Person | Table | Overall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Nets-10 | 0.960 | 0.925 | 0.793 | 0.908 | 0.898 | 0.930 | 0.845 | 0.528 | 0.855 | 0.990 | 0.863 |
| Nets-10 (TTA) | 0.993 | 0.965 | 0.830 | 0.920 | 0.943 | 0.993 | 0.923 | 0.755 | 0.893 | 1.000 | 0.921 |
| Nets-5 | 0.930 | 0.893 | 0.793 | 0.810 | 0.880 | 0.905 | 0.778 | 0.528 | 0.858 | 0.975 | 0.835 |
| Nets-5 (TTA) | 0.983 | 0.950 | 0.830 | 0.823 | 0.925 | 0.973 | 0.875 | 0.755 | 0.893 | 1.000 | 0.901 |

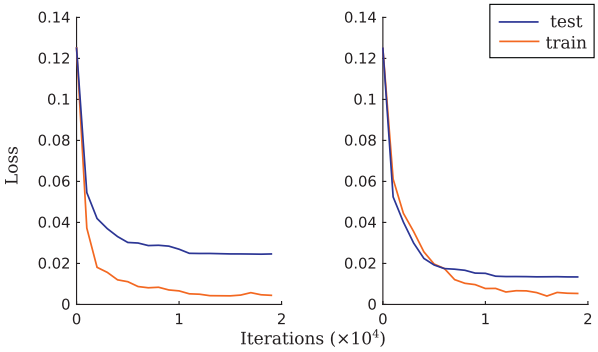


**Fig. 7.** Interference effect. Learning curves are plotted to display the values of loss function in different iterations. Left: trained on inconsistent categories (airplane and person). Right: trained on automatically clustered categories (car, chair, dog and table). Testing loss on consistent categories is apparently lower than that on inconsistent ones.

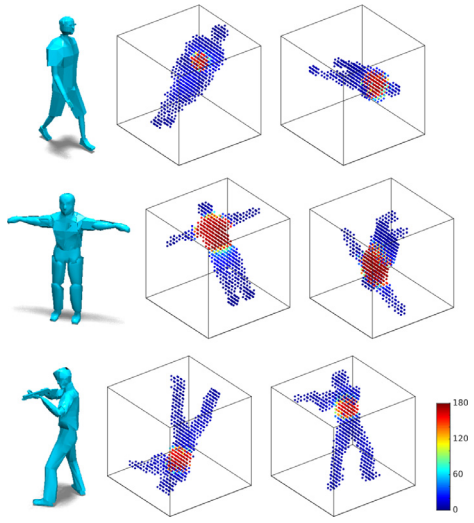

**Fig. 8.** The prediction results will be different if some voxels are hollowed out at different positions. The prediction error (in degrees) after hollowing is illustrated by color mapping. The hotter a voxel is, the more sensitive the network is to the region around it. In these examples, the network always responds strongly to the torsos of human models as the posture and orientation vary, indicating the features extracted by the regression network for person category are orientation- and structure-aware.

two shape categories: airplane and person. The second group (group B) contains four categories: car, chair, dog and table, which were clustered together by our method. The learning process is plotted in Fig. 7. The final training loss of the two groups are similar while the testing loss of group A is apparently higher than that of group B. From the perspective of accuracy, we got 0.713 from the test set of group A and 0.861 from group B, while they were expected to be comparable based on the first row (Nets-10) in Table 3. As a consequence, strong interference effects made the network for group A hard to generalize while its impact on the group clustered by our method is significantly lower. In other words, it is nontrivial to determine the clustering criterion.

**Network visualization.** Inspired by [35], we visualize the network's response to different portions of the voxel grid. We hollowed out a $7 \times 7 \times 7$ cube around each voxel, and computed the angle error between the prediction for the disturbed data and the ground truth to measure the network's sensitivity to the hollowed region. As shown in Fig. 8, the regression network for the person category always responds strongly to the torsos of the human models while shows insensitivity to arms, legs and objects held in hands. These examples demonstrate that this ConvNet has strong ability to learn orientation covariant and posture invariant high-level features. Although upright orientation is the only supervision information, our system is able to locate shape parts with semantic meanings. If more specific labels are available, more semantic and representative features could be learned. A similar example is shown in Fig. 9.

**Generalization capability.** Finally, we present an illustration on how the data-driven method can be used to predict upright orientation for shapes not shown in the training dataset, thus illustrating the generalization ability of the proposed method. As for the examples in Fig. 10, our system classified bird as airplane, piano as table, bed as bathtub and house as cup. The first three cases are predicted correctly while the last one is failed. Our system has generalization ability to some degree, while it would be much better to train new networks for unseen shape categories.

### 4.3. Comparison

Compared with existing approaches, our system can handle more general object categories. The method proposed in Fu et al. [8] is based on the observation that a man-made object should have a supporting base on which it can be steadily positioned and the supporting polygons correspond to faces of the object's convex hull. Nevertheless, this observation would fail on some objects, especially natural ones. Several such examples handled by

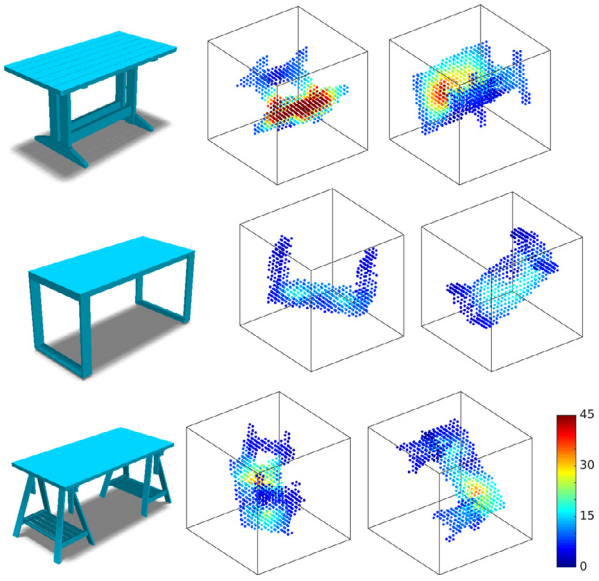


**Fig. 9.** As same as Fig. 8, the prediction error (in degrees) after hollowing is illustrated by color mapping. In these examples, the network is sensitive to the top of table while invariant to the shape of legs.

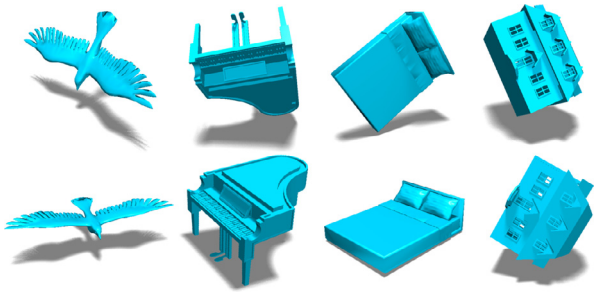

**Fig. 10.** Demonstration of the generalization capability of our method, where we predict upright orientation for shapes not belong to the training categories. The first row shows the models with random orientations. The second row displays the results obtained by our system Nets-5. The first three cases are predicted correctly while the last one is failed.

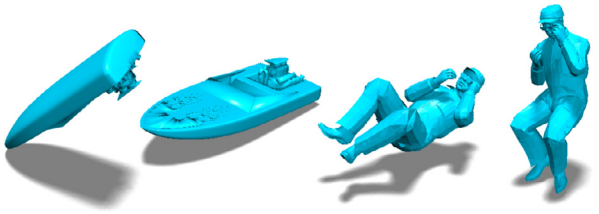

**Fig. 11.** Our method is able to handle shapes whose supporting base is not well defined or do not lie on its convex hull. In each pair, the left one is posed in the random orientation while the right one is posed in the correct orientation predicted by Nets-5. The boat not belonging to our dataset is classified as bathtub.

our method are presented in Fig. 11, demonstrating the advantage of feature learning over feature engineering. The tensor rank minimization approach Wang et al. [31] is not able to deal with shapes with large part not aligned with its upright orientation, as illustrated in Fig. 12. Thanks to

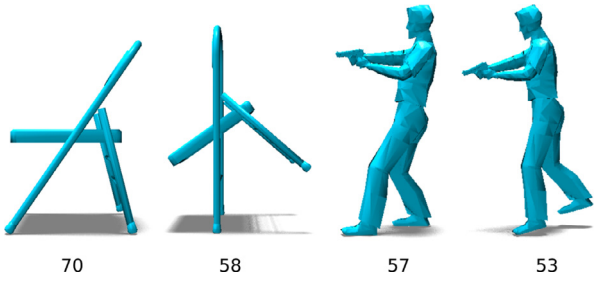


**Fig. 12.** Comparison with Wang et al. [31]. In each pair, the left one, which is posed in its upright orientation predicted by Nets-5, has a higher tensor rank, while the right one posed in incorrect orientation has a lower rank. The rank values are shown below.

**Table 4**
Timing (in seconds) of our system Nets-5 for each test shape. In the last row, each test shape was augmented 10 times and the 10 voxel representations were computed simultaneously in the same mini-batch.

| TTA | Voxelization | Classification | Regression | Total |
|---|---|---|---|---|
| No | 0.008 | 0.014 | 0.012 | 0.034 |
| Yes | 0.078 | 0.033 | 0.039 | 0.150 |

the learning ability of ConvNets, these objects can be handled by our method correctly.

Our approach also has an advantage of efficiency over other techniques. The method proposed by Fu et al. [8] contains two main steps: convex hull computation for candidate supporting bases selection and feature extraction for candidates evaluation. These two steps took 5 s on average for each object. In Wang et al. [31], the tensor rank minimization problem is highly nonlinear and hard to optimize. Therefore a genetic algorithm is adopted which took about 1–2 min for each shape. In contrast, our method reached a much more fast speed due to the parallel nature of ConvNets which is match for GPU acceleration. Furthermore, a batch of data can be processed simultaneously. The detailed timing results are listed in Table 4, from which we can conclude that our method is at least 30 times faster than existing approaches.

## 5. Conclusions and future work

We proposed a data-driven method for 3D object upright orientation estimation using 3D Convolutional Networks. Thanks to the feature learning ability of ConvNets, not only man-made objects but also natural ones can be handled. In addition, a distance based clustering technique was proposed to reduce the memory costs and a test-time augmentation procedure was proposed to further improve the accuracy. The experimental results demonstrate the efficiency and effectiveness of our approach. Besides this, the visualization results indicate that ConvNets are able to capture more semantic features if more informative labels are provided. At last, our method is extremely efficient, so it can be used as preprocessing to speed up several geometry processing tasks, such as 3D shape retrieval, matching and registration.

On the other hand, our method can still be improved in several directions. First, this approach is not as accurate as geometric methods. We consider improving the performance by geometric technique such as finding supporting bases (if available) around our result. Second, techniques of committee machines [12] should be considered to optimize the entire system all together, other than training the networks for classification and regression independently. Third, further visualization [35] works should be done to gain more insights from the trained networks and to answer several mysterious questions, such as: Why and how do the networks work? Why does the network trained on airplane work well on bathtub and vice versa? Last and not least, we would like to adopt our system to take range image as input for robotics tasks.

## Acknowledgments

We would like to acknowledge Hao Li from University of Science and Technology of China and Hongxuan Zhang from the Pennsylvania State University for their helpful suggestions on this paper. This work was supported by the NSF of China (nos. 61303148, 61222206, 11526212, 11426236), NSF of Anhui Province, China (no. 1408085QF119), Specialized Research Fund for the Doctoral Program of Higher Education under contract (no. 20133402120002) and the Hundred Talents Program of the Chinese Academy of Sciences.

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.gmod.2016.03.001.

## References

[1] S. Abbasi, F. Mokhtarian, Automatic view selection in multi-view object author recognition, in: 15th International Conference on Pattern Recognition, 2000. Proceedings., 1, 2000, pp. 13–16.
[2] S. Biasotti, S. Marini, M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, 3D shape matching through topological structures, in: Discrete Geometry for Computer Imagery, 2003, pp. 194–203.
[3] G. Ciocca, C. Cusano, R. Schettini, Image orientation detection using LBP-based features and logistic regression, Multimedia Tools Appl. 74 (9) (2013) 3013–3034.
[4] S. Dieleman, Classifying Plankton with Deep Neural Networks, 2015, http://benanne.github.io/2015/03/17/plankton.html (accessed 03.09.15).
[5] A. Dosovitskiy, J. Tobias Springenberg, T. Brox, Learning to generate chairs with convolutional neural networks, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[6] M. Feixas, M. Sbert, F. González, A unified information-theoretic framework for viewpoint selection and mesh saliency, ACM Trans. Appl. Percept. (TAP) 6 (1) (2009) 1.
[7] W. Feng, Z. Yang, J. Deng, Moving multiple curves/surfaces approximation of mixed point clouds, Commun. Math. Stat. 2 (1) (2014) 107–124.
[8] H. Fu, D. Cohen-Or, G. Dror, A. Sheffer, Upright orientation of man–made objects, ACM Trans. Graphics (TOG) 27 (3) (2008) 42:1–42:7.
[9] S. Gupta, R. Girshick, P. Arbeláez, J. Malik, Learning rich features from RGB-D images for object detection and segmentation, in: Computer Vision—ECCV 2014, 2014, pp. 345–360.
[10] R. Hartley, J. Trumpf, Y. Dai, H. Li, Rotation averaging, Int. J. Comput. Vis. 103 (3) (2013) 267–305.
[11] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, second, Springer, 2009.
[12] S. Haykin, Neural Networks: A Comprehensive Foundation, second, Prentice Hall PTR, 1998.
[13] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, K. Ramani, Three-dimensional shape searching: state-of-the-art review and future trends, Comput. Aided Des. 37 (5) (2005) 509–530.
[14] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.
[15] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Machine Intell. 35 (1) (2013) 221–231.
[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the ACM International Conference on Multimedia, 2014, pp. 675–678.
[17] Y. Jiang, M. Lim, C. Zheng, A. Saxena, Learning to place new objects in a scene, Int. J. Robotics Res. 31 (9) (2012) 1021–1043.
[18] Y. Jin, Q. Wu, L. Liu, Unsupervised upright orientation of man-made models, Graphical Models 74 (4) (2012) 99–108.
[19] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation invariant spherical Harmonic representation of 3D shape descriptors, in: Eurographics Symposium on Geometry Processing, 2003.
[20] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[21] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, 2012, pp. 9–48.
[22] C.-K. Lin, W.-K. Tai, Automatic upright orientation and good view recognition for 3d man-made models, Pattern Recognit. 45 (4) (2012) 1524–1530.
[23] A. Lumini, L. Nanni, Detector of image orientation based on Borda count, Pattern Recognit. Lett. 27 (3) (2006) 180–186.
[24] J. Luo, M. Boutell, Automatic image orientation detection via confidence-based integration of low-level and semantic cues, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 715–726.
[25] D. Plemenos, M. Benayada, Intelligent display in scene modeling. New techniques to automatically compute good views, in: International Conference on Computer Graphics & Vision, 1996.
[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[27] H. Su, S. Maji, E. Kalogerakis, E.G. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: Proceedings ICCV, 2015.
[28] J.W. Tangelder, R.C. Veltkamp, A survey of content based 3D shape retrieval methods, Multimedia Tools Appl. 39 (3) (2008) 441–471.
[29] P.-P. Vázquez, M. Feixas, M. Sbert, W. Heidrich, Automatic view selection using viewpoint entropy and its application to image-based modelling, Comput. Graphics Forum 22 (4) (2003) 689–700.
[30] P.-P. Vázquez, E. Monclús, I. Navazo, Representative views and paths for volume models, in: Smart Graphics, 2008, pp. 106–117.
[31] W. Wang, X. Liu, L. Liu, Upright orientation of 3d shapes via tensor rank minimization, J. Mech. Sci. Technol. 28 (7) (2014) 2469–2477.
[32] Y.M. Wang, H. Zhang, Detecting image orientation based on low-level visual content, Comput. Vis. Image Understand. 93 (3) (2004) 328–346.
[33] Z. Wu, S. Song, A. Khosla, L. Zhang, X. Tang, J. Xiao, 3D shapenets: a deep representation for volumetric shape modeling, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[34] K. Xu, H. Zhang, D. Cohen-Or, B. Chen, Fit and diverse: set evolution for inspiring 3D shape galleries, ACM Trans. Graphics (TOG) 31 (4) (2012) 57:1–57:10.
[35] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision–ECCV 2014, 2014, pp. 818–833.
[36] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, Int. J. Comput. Vis. 13 (2) (1994) 119–152.
[37] Z. Zhang, A. Ganesh, X. Liang, Y. Ma, Tilt: transform invariant low-rank textures, Int. J. Comput. Vis. 99 (1) (2012) 1–24.
[38] Z. Zhu, X. Wang, S. Bai, C. Yao, X. Bai, Deep learning representation using autoencoder for 3D shape retrieval, in: 2014 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), 2014, pp. 279–284.