DOI: 10.1111/cgf.14419
Pacific Graphics 2021
E. Eisemann, K. Singh, and F.-L Zhang
(Guest Editors)

Volume 40 (2021), Number 7

UprightRL: Upright Orientation Estimation of 3D Shapes via Reinforcement Learning

Luanmin Chen[†1] Juzhan Xu[†1] Chuan Wang[1] Haibin Huang[2] Hui Huang[1] Ruizhen Hu[‡1]

[1]Shenzhen University [2]Kuaishou Technology

Figure 1: *Grasp and place the object to its upright orientation guided by our UprightRL model.*

Abstract

In this paper, we study the problem of 3D shape upright orientation estimation from the perspective of reinforcement learning, i.e. we teach a machine (agent) to orientate 3D shapes step by step to upright given its current observation. Unlike previous methods, we take this problem as a sequential decision-making process instead of a strong supervised learning problem. To achieve this, we propose UprightRL, a deep network architecture designed for upright orientation estimation. UprightRL mainly consists of two submodules: an Actor module and a Critic module which can be learned with a reinforcement learning manner. Specifically, the Actor module selects an action from the action space to perform a point cloud transformation and obtain the new point cloud for the next environment state, while the Critic module evaluates the strategy and guides the Actor to choose the next stage action. Moreover, we design a reward function that encourages the agent to select action which is conducive to orient model towards upright orientation with a positive reward and negative otherwise. We conducted extensive experiments to demonstrate the effectiveness of the proposed model, and experimental results show that our network outperforms the state-of-the-art. We also apply our method to the robot grasping-and-placing experiment, to reveal the practicability of our method.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ *Reinforcement learning; Sequential decision-making process; Upright orientation;*

1. Introduction

In this paper, we tackle the problem of 3D shape upright orientation estimation. This is a long standing problem in both computer graphics as well as vision. Orientating 3D shape into its upright is an important step for certain applications, such as shape retrieval, model recognition, segmentation and so on. However, a model is usually acquired in poor conditions with its orientation incorrectly placed, and manually correcting these models in software like Maya, 3DS Max or Meshlab by human is a time-consuming, error-prone and tedious task. To this end, an automatic, robust and efficient upright orientation estimation approach is urgently needed nowadays.

Up to now, there have been several research works on 3D shape upright orientation estimation and achieve certain results, including methods based on either hand-crafted features or neural net-

† Joint first authors
‡ Corresponding author: Ruizhen Hu (ruizhen.hu@gmail.com)

works. The first work was proposed by Fu *et al.* [FCDS08] motivated by their observation that 3D models generated from various modeling tools or scanning systems tend to have different upright orientations. They developed a two-stage method where candidate bases were proposed and then evaluated based on their geometry properties. Following [FCDS08], Jin *et al.* [JWL12] and Wang *et al.* [WLL14] proposed matrix rank minimization and 3D tensor minimization methods to align models with coordinate axes, and then selected the final base from candidate bases according to the geometrical features proposed by Fu *et al.* [FCDS08]. Meanwhile, with the fast development of deep neural network (DNN), Liu *et al.* [LZL16] solve the upright orientation estimation problem by first determining the object category and then utilizing a corresponding regression network to fit the upright direction. Nonetheless, these methods based on hand-crafted features or networks have certain limitations, due to either strict requirement on data input such that the object demands a supported base for its standing, or low efficiency as different regression networks need to be trained for different object categories.

In recent years, reinforcement learning reveals its advantages over supervised learning in some specific aspects, by converting the original problem as a sequential decision process [ADBB17]. For example, in the image cropping task, compared with the time-consuming and storage-intensive sliding window method that leads to a large number of candidate regions, [LWZH18] demonstrates its better performance by utilizing reinforcement learning and selecting actions from a limited action space set only, which results in low time and space complexity. These works inspire us to propose a reinforcement learning based solution to the shape upright orientation estimation problem.

There are two main challenges for the adaption of reinforcement learning for the upright orientation task: how to efficiently extract environmental observation from the agent's view given a point cloud input as well as how to evaluate the action space based on the current state. Towards this end, we propose our network, namely UprightRL, which first extracts features from the input cloud and outputs the environmental observation for the agent, and then the agent makes strategy decision accordingly and evaluate the current strategy. The whole process is achieved by an Actor-Critic algorithm [MBM*16]. The Actor branch conducts strategy selection from an action space. Once the action is selected, the Actor branch performs a rotation on the current state point cloud to obtain a new state. The Critic branch then evaluates the current strategy with a reward designed as feedback after the evaluation, which indicates a beneficial action by a positive value or a damaging one otherwise. With the network training continues, the strategy selection becomes increasingly better and the evaluator works in a gradually more accurate manner. The network finally converges with the accumulated reward being continuously increased until unchanged, so as to achieve the upright orientation task.

To demonstrate the significance of our method, we conduct quantitative and qualitative comparisons with a network without reinforcement learning and the state-of-the-art methods. The last but not the least, to verify the practicability, we apply it to the robot grasping-and-placing simulated experiment to improve the robot operation quality because it can filter the unreasonable grasp poses.

- We cast the 3D shape upright orientation estimation problem as a sequential decision making problem, and propose a reinforcement learning architecture UprightRL network to solve it step by step.
- We adapt the Actor-Critic algorithm for our problem formulation and customize a reward function which induces the network gradually forward shape upright orientation estimation.
- We conduct extensive comparison experiments to demonstrate the superiority of the proposed reinforcement learning model over the state-of-the-art method and other alternative baseline based on supervised learning.
- We further apply our method to robot grasping-and-placing experiment to the clarify our method's utility in practice. The experimental result shows the upright orientation predicted by our method can filter out unreasonable grasping poses, making the robot place object better.

2. Related Work

2.1. Upright Orientation Estimation Methods

Hand-crafted feature based methods. In computer graphics, there have existed several solutions proposed to estimate the upright orientation for given objects. Fu *et al.* [FCDS08] used the supported candidate bases, from which a base is selected as the solution if the 3D shape can stand stably on it. This method is based on supervised learning and is feasible for most man-made models but would fail for objects with supported bases that are hard to well define, such as boat.

Besides the aforementioned method, Jin *et al.* [JWL12] proposed an iterative optimization algorithm with an unsupervised learning strategy. It aligns the 3D shape with coordinate axes based on the observation that a 3D shape in its upright orientation should have a low rank projection matrix. Wang *et al.* [WLL14] follows the similar principle, while the guidance becomes the 3D tensors composed by the shape rather than the projection matrix. By minimizing the rank of the 3D tensors via a genetic algorithm, a local optimum and six candidate supported bases are obtained, so as to achieve orientating the 3D shape by combining stability, visibility and symmetry properties proposed in [FCDS08] to select the optimal base. It is worth-noting that, as unsupervised methods, these two methods are able to handle a wider variety of shape types as they involve neither model training nor prior knowledge, but they are limited to shapes with rough symmetry.

To sum up, these hand-crafted feature based methods are limited by either the input data types or the constraint of the output results, so they need to be further improved.

DNN based methods. With the development of DNN in recent years, there are increasingly more tasks that apply DNN as a solution, for example the intensive usage of Convolutional Neural Network (CNN) in image classification. With this inspiration, Liu *et al.* [LZL16] proposed a DNN architecture to solve upright orientation estimation of the 3D shapes. They classify the voxelized shapes by 3D CNN followed by regressing the shapes' upright orientation according to each category regression model. As an end-to-end

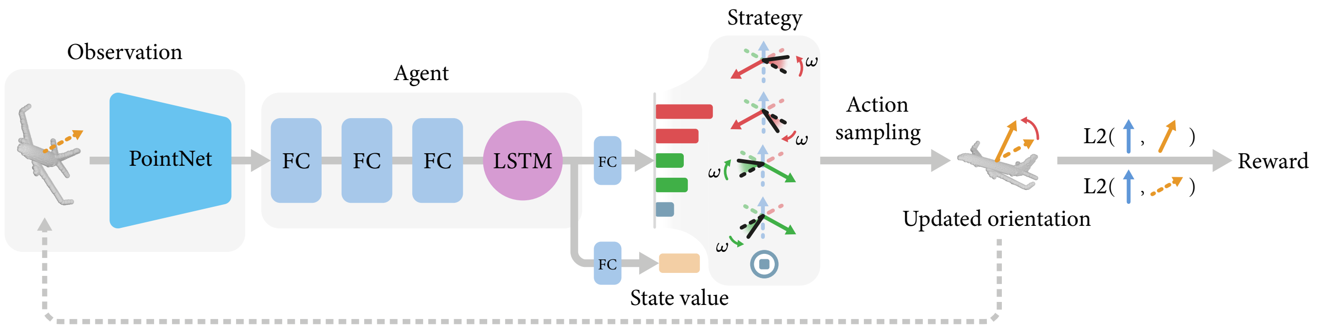

Figure 2: *The network architecture of UprightRL. Given a 3D shape point cloud at time step t, the PointNet module extracts its feature and then passes it to the Actor-Critic module, so as to obtain two outputs. Specifically, the Actor branch samples a single action from an action space at each time step to transform the point cloud, and the Critic branch is responsible for estimating the reward expectation at the current state. The reward is set to a positive value if the L2 difference between the shape's orientation vector and its ground truth is reduced compared with that in the last time step, and to a negative value otherwise. The agent is composed of three fully connected (FC) layers and one LSTM layer. The agent extracts features and feeds to the FC layers of two subsequent branches, where a actor branch produces 5-dimensional vectors as actions and a critic branch outputs scalar value as state value.*

learning approach, it is able to handle most models but it involves multiple models to be regressed after the classification for different classes. If the number of classes and the variation between classes are large, training a plenty of models becomes a time-consuming and storage-intensive task.

2.2. Reinforcement Learning

During the development of deep supervised and unsupervised learning, deep reinforcement learning (DRL) also makes a significant influence on robots or games for their advanced applications, which are gradually applied to computer vision and other fields and show its superiority than more traditional methods.

Li *et al.* [LWZH18] first brought DRL to solve the image cropping problem. Compared with the previous sliding window based method, it has the advantage that does not involve proposal of tens of thousands of region candidates, so as to save lots of time. It is also able to have cropping of flexible height-width ratios as finite actions, so as to complete the task in finite steps, producing cropped images in arbitrary shapes. For combinatorial optimization, Bello *et al.* [BPL*17] incorporated DRL into Pointer Network [VFJ15], which is a well known neural architecture for tackling combinatorial problems, to solve the Travelling Salesman Problem. Based on the framework of Bello *et al.* [BPL*17], Hu *et al.* [HZY*17] applied DRL to solve the 3D packing problem. Both of these problems are NP-hard combinatorial problems with extremely large search space, and their works demonstrate the generalization of supervised data is rather poor compared to an RL agent which explores different cases and observes their corresponding rewards. Moreover, Park *et al.* [PHBD21] presented two methods based on DRL for adaptive streaming of 360-degree videos. Unlike prior works that use pre-determined rules for rate adaptation, they can dynamically determine which tiles to download at what qualities and when, depending on the network conditions, which has superiority than state-of-the-art 360-degree tiled video streaming techniques. While for unsupervised transfer learning task, Zhang *et al.* [ZYD21] proposed a novel framework called adversarial reinforcement learning (ARL) for domain adaptation. Reinforcement learning is employed as a feature selector to identify the closest feature pair between source and target domains, which mitigate the discrepancy between these two domains and their performance is better than the state-of-the-art domain adaptation methods. However, there is limited research work that applies reinforcement learning to the task of object upright orientation estimation. There exists sufficient challenges rather than a trivial application, including how to efficiently extract environmental observation given a point cloud and how to evaluate the action space based on the current state. Towards this end, we specifically designed action space, rewards function and training strategy based on the Actor-Critic algorithm for the challenging task of upright orientation estimation. These customized settings make the application of reinforcement learning to upright orientation estimation possible.

3. Algorithm

We formulate upright orientation estimation of 3D shapes as a sequential decision-making process, during which an agent interacts with the environment, and takes a series of actions to optimize the strategy so as to achieve orientation estimation. The proposed network UprightRL is illustrated in Figure 2, which consists of a point cloud feature extractor and an Actor-Critic module. A reward function is defined to guide the optimization of this reinforcement learning problem. Below, we introduce each component of UprightRL, including observation state, action space, the reward function as well as the implementation details of network structure.

3.1. Observation States and Action Space

Observation state. Given a point cloud state $\mathcal{P}_t$ at time step t, we extract its feature o_t serving as the observation to the agent by a feature extractor $f(\cdot)$, i.e.

$$o_t = f(\mathcal{P}_t) \tag{1}$$

The current observation state is further defined as all the observations up to time t, i.e. $s_t = \{o_0, o_1, ..., o_t\}$. The agent makes decisions based on this state by taking all the observations into account, which is similar to human's decision-making mechanism.

Action space. The action space embodies five actions that can be divided into three groups: rotate around x-axis, rotate around y-axis, and stop. The first two groups aim to rotate the 3D shape in upright orientation without caring about any z-axis related rotation, because the shape can achieve its upright pose with x-axis and y-axis rotations only. In each group of x-axis and y-axis rotation, it involves clockwise and counterclockwise rotation at a fixed angle respectively. The stop action is a termination trigger for the agent, that is, once this action is taken, the agent will stop rotation and output the current point cloud with its 3D orientation vector as the upright orientation. In theory, the agent should be able to fully cover the data space of random orientation of the input point clouds. The observations and action space can be illustrated in Figure 2.

3.2. Reward Function

We further define a reward function to guide the optimization of our UprightRL network, with a basic capability to reflect how good is an action the agent takes. Intuitively, if the difference between the orientation predicted in current time step and the ground truth becomes smaller than that obtained in the last step, we consider the orientation task is being solved in the correct track. As such, we define the reward function as follows,

$$r_t = b \cdot Sign(\mathcal{M}_t - \mathcal{M}_{t+1}) - \mu \cdot (t+1) \quad (2)$$

where $\mathcal{M}_t$ is the L_2 difference between the current orientation and the ground truth, i.e.

$$\mathcal{M}_t = \left\| \overrightarrow{\mathcal{U}}^{\text{gt}} - \overrightarrow{\mathcal{U}}_t^{\text{pre}} \right\|_2 \quad (3)$$

$Sign(\cdot)$ is the sign function, and b equals to 5 and 1 if the sign of $(\mathcal{M}_t - \mathcal{M}_{t+1})$ is positive and negative respectively. Here we use the sign function instead of the original $(\mathcal{M}_t - \mathcal{M}_{t+1})$ to constrain the range of the reward value, because we observed a more stable and easier convergence could be achieved in practice. In order to speed up the rotation process, we add an extra negative penalty term weighted by a constant μ, set as 0.001 in our experiments. This term will cause lower reward for the agent after a large number of steps are preformed.

3.3. The UprightRL Model

With the observation state, action space and reward function being defined as above, our UprightRL network is presented as follows. An illustration of it is shown in Figure 2, where a random rotated point cloud $\mathcal{P}$ is given as input, and the network produces an upright orientated version of it as output, i.e. $\mathcal{P}_{out}$. We utilize PointNet [QSMG17] as the point cloud feature extractor $f(\cdot)$, which consists of three convolutional and three fully connected (FC) layers. We use PointNet over PointNet++ [QYSG17] as it is a light-weighted backbone and is more efficient and memory-friendly for training. An 1024-D observation o_t is fed to an Actor-Critic module, where an agent being composed of three FC layers and one LSTM layer is used to fuse o_t with all the historical observations $\{o_0, o_1, ..., o_{t-1}\}$. The Actor-Critic module then produces two branches of strategy output. One is a 5D vector with each entry of it indicating an action probability, and the other is the expectation of the current accumulative return that evaluates the current strategy. According to the strategy output, the agent samples relative action and transforms the point cloud $\mathcal{P}$ and its orientation vector. The forward pass of the network continues until a stop action is sampled or the maximum steps threshold is reached.

Algorithm 1: Training procedure of the UprightRL model

```
Input: Original rotated randomly point clouds $\mathcal{P}$
Output: Predicted point clouds $\mathcal{P}_{out}$ in upright orientation
  Initialize $\mathcal{P}_0 \leftarrow \mathcal{P}, t \leftarrow 0$
  repeat
    $t_{\text{start}} \leftarrow t, d\theta \leftarrow 0, d\theta_v \leftarrow 0$
    repeat
      $o_t \leftarrow f(\mathcal{P}_t)$, where $f(\cdot)$ is a feature extractor.
      $s_t \leftarrow LSTM_{AC}(o_t)$
      $\mathcal{P}_{t+1}, \overrightarrow{\mathcal{U}}_{t+1}^{\text{pre}} \leftarrow Transformation\left(\mathcal{P}_t, \overrightarrow{\mathcal{U}}_t^{\text{pre}}; \pi(a_t|s_t;\theta)\right)$
      $r_t \leftarrow Reward\left(\overrightarrow{\mathcal{U}}_t^{\text{pre}}, \overrightarrow{\mathcal{U}}_{t+1}^{\text{pre}}, t\right)$
      $t \leftarrow t+1$
    until $t - t_{start} = t_{max}$ or $a_{t-1}$ is the termination action;
    $R = \begin{cases} 0 & \text{if } a_{t-1} \text{ is the termination action} \\ V(s_t;\theta_t) & \text{for other actions} \end{cases}$
    for $i \in \{t-1, \ldots, t_{\text{start}}\}$ do
      $R \leftarrow r_i + \gamma R$
      $d\theta \leftarrow$
      $d\theta + \nabla_\theta log\pi(a_i|s_i;\theta)(R - V(s_i;\theta_v)) + \beta\nabla_\theta H(\pi(s_i;\theta))$
      $d\theta_v \leftarrow d\theta_v + \nabla_{\theta_v}(R - V(s_i;\theta_v))^2/2$
    end for
    Update $\theta$ with $d\theta$ and $\theta_v$ with $d\theta_v$
  until $t = t_{max}$ or $a_{t-1}$ is the termination action;
```

Optimization. To train the proposed Upright model, we follow the classical actor-critic task solutions. Our model training procedure is detailed in Algorithm 1. Specifically, we define the accumulative return R_t at time step t as follows,

$$R_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) \quad (4)$$

where γ is a discount factor, r_t is the reward at step t, $V(s_t;\theta_v)$ is the value output under state s_t. θ_v represents the network parameters of the critic branch, and k ranges from 0 to t_{max}, where t_{max} is the maximum number of steps.

We further apply advantage function [MBM*16] and entropy normalization [WP91] to describe the optimization term of strategy output, aiming to maximize the advantage function $R_t - V(s_t;\theta_v)$ and the entropy $H(\pi(s_t;\theta))$, where $\pi(s_t;\theta)$ is a probability distribution of actions in the strategy outputs, θ is the parameters of Actor branch network, and $H(\cdot)$ is the entropy function, which aims to increase the diversity of action selection, so as to enable the agent to make more flexible decision. Moreover, the optimization goal of value output is minimizing $(R_t - V(s_t;\theta_v))^2/2$. The gradient of the actor branch can be formulated as $\nabla_\theta log\pi(a_t|s_t;\theta)(R_t - V(s_t;\theta_v)) + \beta\nabla_\theta H(\pi(s_t;\theta))$, where β is the weight of entropy, and the gradients of the critic branch would be $\nabla_{\theta_v}(R_t - V(s_t;\theta_v))^2/2$.

4. Experiments

To demonstrate the effectiveness of our proposed UprightRL, we conduct experiments on several datasets, including complete point cloud, partial point cloud as well as single scan as input. We also compare UprightRL with a BaseNet which is trained as a supervised regression network, and also with previous state-of-the-art methods. Both qualitative and quantitative results show the superior performance of UprightRL.

4.1. Dataset and Implementation Details

Complete and partial point cloud dataset. We use the complete and partial point clouds provided by Completion3D Point Clouds Benchmark [TKR*19]. It contains 8 categories with around 30,000 shapes. Specifically, the complete point clouds are uniformly sampled on the mesh surfaces and the partial point clouds are back-projected from 2.5D depth images [YKH*18], which aims to simulate the real scanning partial data. We further downsample the point clouds into consistent input size (2048 for our experiments).

Single scan dataset. To fully evaluate UprightRL, we also generate a synthetic single scan dataset. Specifically, we use scanning tools similar to [BRHS14; YSGG17], which were designed to simulate Kinect scanning point clouds generation. For each scan, we take one 3D model and align its mass center to the origin. We then apply a random rotation on the mesh and generate the simulated scan from a fixed view point, i.e. $(0,0,1.5)$ in our experiments.

Implementations: We split these datasets into 80%/10%/10% for training/testing/validation respectively. Our method is implemented using the PyTorch deep learning library [PGM*19]. The UprightRL model is trained with a single NVIDIA® ITITAN XP GPU, where one epoch training takes about 23 minutes with around 250 epochs for convergence. We set maximum steps $t_{max} = 20$, and discount factor of rewards $\gamma = 0.4$ for the accumulative returns function. The entropy weight β, strategy weight, value weight are set as $\{0.1, 1.0, 1.0\}$ respectively for our experiments. Adam [KB14] optimizer with learning rate 10^{-3} is utilized to optimize the network.

In our experiments, the rotation angle step is set as $\omega = 4^\circ$, which brings us the best performance over rotation angles from 1° to 10°, considering a good trade-off between accuracy and time steps, as shown in Figure 3.

4.2. Comparison with Existing Methods

We first quantitatively compare our method with a supervised learning baseline and the state-of-the-art approach [LZL16], and then show more qualitative results of our method in the end.

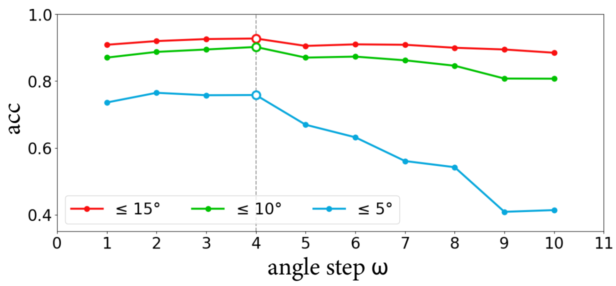

Figure 3: *Rotation angle step setting with the best performance achieved when $\omega = 4^\circ$.*

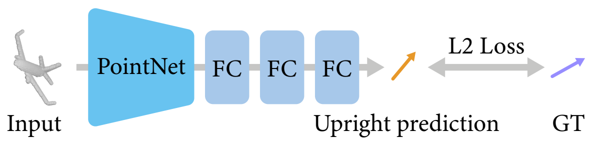

Figure 4: *The architecture of the BaseNet. Instead of using a sequential decision-making process powered by reinforcement learning, a simple supervised network is designed for comparison.*

Evaluation Metric. We follow the metric used in [LZL16] by first computing the angle error between the predicted orientations and their ground truths and then making statistics on these errors, calculating the percentage of samples within 15°, 10°, 5° error respectively. The angle error can be easily obtained via trigonometric function:

$$\left\langle \vec{\mathcal{U}}^{\text{pre}}, \vec{\mathcal{U}}^{\text{gt}} \right\rangle = \left| \arccos\left(\vec{\mathcal{U}^{\text{pre}}} \cdot \vec{\mathcal{U}^{\text{gt}}} \right) \right| \cdot \frac{180^\circ}{\pi} \tag{5}$$

Note that in [LZL16], only 15° is used as they believe this error threshold is enough for most cases. We extend the error threshold to 10° and 5° to evaluate the results in a more accurate manner, to reveal the superiority of our UprightRL method.

4.2.1. Ours vs. supervised learning baseline

To verify the effectiveness of our model powered by reinforcement learning, we first simplify our model into a network BaseNet for supervised learning. As shown in Figure 4, a upright vector of each randomly rotated point clouds is predicted without the sequential decision making process. We use the PointNet [QSMG17] encoder to extract the features, which are then fed into the multiple fully connected layers, gradually reducing the dimensions to produce the upright orientation vector prediction as the output. The loss of BaseNet is the L2 norm between the prediction and the ground truth orientation vectors. That is,

$$L_{\text{up\_base}} = \left\| \vec{\mathcal{U}}^{\text{pre}} - \vec{\mathcal{U}}^{\text{gt}} \right\|_2 \tag{6}$$

In order to ensure the fairness of the comparison, we train and test the BaseNet and UprightRL models on the aforementioned three datasets. Thanks to UprightRL model's sequential making-decision process,the predicted orientation can achieve closer to

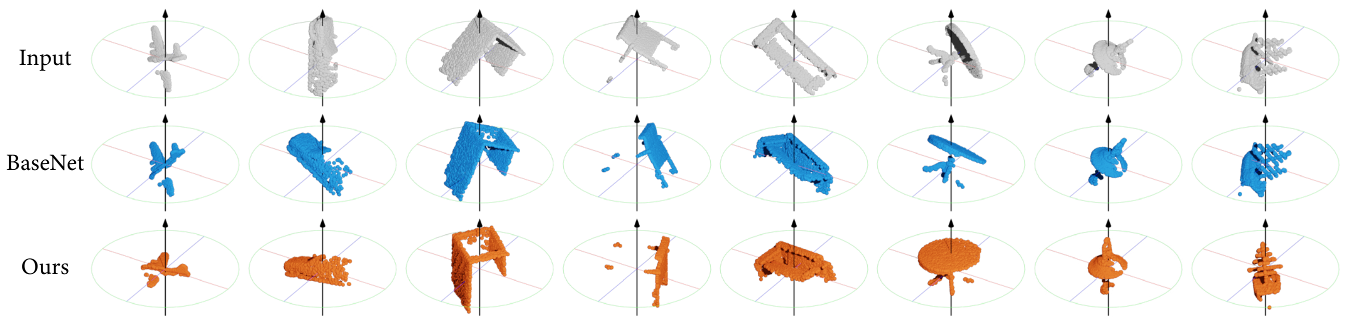

Figure 5: *Qualitative comparison results with BaseNet on the single scan dataset. The first row shows the input point cloud, the second row shows the results obtained using BaseNet, and the last row shows our results.*

Table 1: *Comparison between our UprightRL model and BaseNet on the Complete, Partial, Single scan datasets.*

Dataset	Method	$\leq 15^\circ$	$\leq 10^\circ$	$\leq 5^\circ$
Complete	BaseNet	0.793	0.690	0.418
	Ours	**0.940**	**0.927**	**0.836**
Partial	BaseNet	0.609	0.476	0.238
	Ours	**0.881**	**0.851**	**0.699**
Single scan	BaseNet	0.783	0.649	0.337
	Ours	**0.927**	**0.902**	**0.758**

the ground truth. As seen in Table 1, the percentage of angle errors within 5° of UprightRL is around 2 times higher than that of BaseNet, revealing the better performance of our model. It is also noted that despite of the shapes being complete or partial during training, our UprightRL model has a consistent better performance. It is also worth-noting that with the completeness of the data drops, the percentage of angle error within 5° for UprightRL drops less than that for BaseNet, i.e. 13.7% from 0.836 to 0.699 vs. 43.1% from 0.418 to 0.238, which shows the superior robustness of UprightRL method.

We also show some representative results from the single scan dataset in Figure 5 for qualitative comparison, and it can be seen that our method generally produces more accurate results than BaseNet. For example, the airplane (col 1) and the car (col 2) point clouds are vertical to their upright states. In these cases, our model performs sequential actions on them to get the optimal upright orientation. As for the box-like category cabinet (col 3), although the point clouds are partial, our model still predicts its upright orientation correctly. In the example of chair (col 4) which has back and two legs only, our model consistently obtain the correct upright orientation and make the legs standing on straightly. The similar case also applies to the couch (col 5) and the table (col 6) example. For the lamp (col 7) and vessel (col 8), they have non-flat bases as the airplane, which brings too much confusion to the BaseNet. In contrast, our model produces accurate and stable results for all the challenging examples, thanks to the sequential making-decision process that fuses not only the current information but also the historical experience.

Table 2: *Comparison between our UprightRL model and previous work [LZL16] on their dataset. Note TTA means test-time augmentation.*

Data	[LZL16]	[LZL16]-TTA	**Ours**	**Ours-TTA**
Airplane	0.960	**0.993**	0.963	0.990
Bathtub	0.925	0.965	0.955	**1.000**
Bicycle	0.793	0.830	0.955	**0.978**
Car	0.908	0.920	0.968	**0.998**
Chair	0.898	0.943	0.938	**0.950**
Cup	0.930	**0.993**	0.938	0.960
Dog	0.845	0.923	0.908	**0.980**
Fruit	0.528	0.755	0.940	**0.988**
Person	0.855	0.893	0.865	**0.913**
Table	0.990	**1.000**	0.975	**1.000**
Average	0.863	0.922	0.940	**0.976**

4.2.2. Ours vs. state-of-the-art method

We further compare our method with a state-of-the-art one [LZL16] on their dataset in ModelNet since their method cannot be directly applied to the point cloud datasets as our method. To make the comparison fair enough, we strictly follow the training and test data split in [LZL16]. The only modification is that our input data is point cloud sampled from the original mesh models, while their input is voxelized data. As for the data augmentation, [LZL16] involves random rotation within 360° around the orientation axis, making the ground truth orientation distributed on a unit sphere, while the random rotation angle for our method is within 180° only, as our action space possessing the rotation actions are around the Cartesian axes clockwisely or counterclockwisely. In other words, we reduce the ambiguous expression of the same shape state before data input.

We follow the same metric as in [LZL16] to calculate percentage of angle errors within 15°. As seen in columns 2 and 4 in Table 2, the accuracy of our UprightRL model is 94%, i.e. 7.7% higher than that of [LZL16]. As for the 10 categories of the input shapes, the prediction accuracy of our method is consistently higher than 86.5%, while [LZL16] only has 6 categories above it.

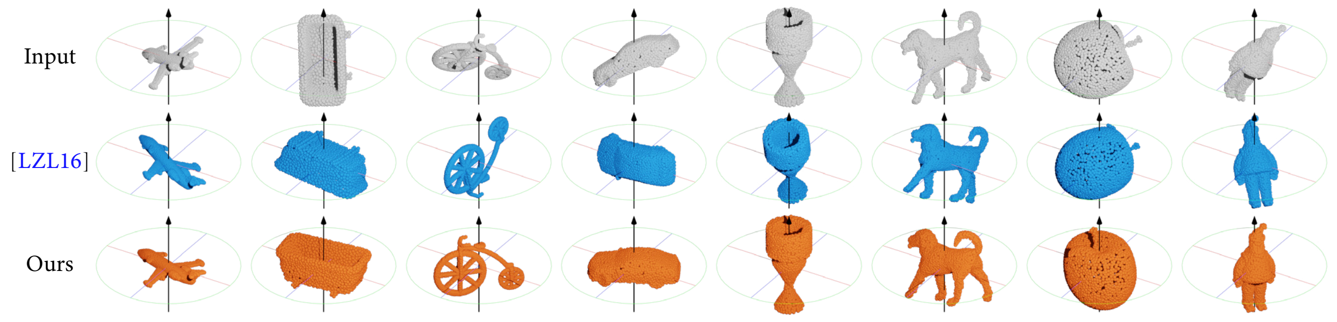

Figure 6: *Qualitative comparison results with [LZL16] on their dataset. The first row shows the input point cloud, the second row shows the results obtained using the method of [LZL16], and the last row shows our results.*

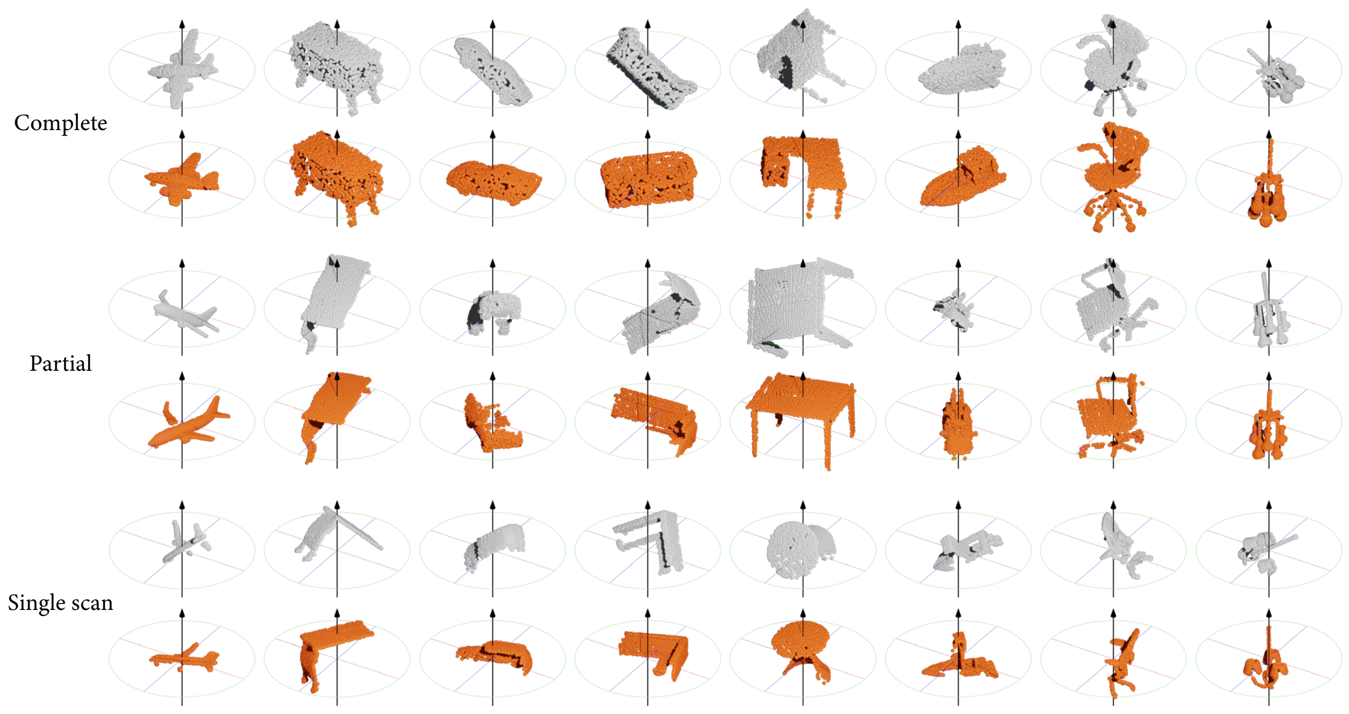

Figure 7: *Some representative results obtained by our method on input point cloud with different completeness. The top two rows show the input and corresponding results on the complete point cloud dataset, the middle two row are those for partial point cloud dataset, and the last two rows are those for single scan dataset.*

Our method only fails by a marginal accuracy (0.015) in the table category compared with [LZL16], but produces significant higher percentage in the other 9 categories, especially for the fruit with 92% against 52.8%. Besides, we also apply test-time augmentation (TTA) to the evaluation as [LZL16], that is taking average of the output by sampling multiple times. As seen in columns 3 and 5, with TTA involved, the accuracy by our method is further improved, and the superiority of our method is still kept in most categories and in the overall dataset.

Figure 6 further shows some qualitative results by our method and [LZL16]. It can be clearly observed that our method has more accurate results. In the first four columns, Liu *et al.* [LZL16] may get opposite or vertical orientation. In the next four columns, the results have small angles with the upright orientation vectors, but [LZL16] fails to capture this difference and keeps the errors in the output. More interestingly, the cup has been already in its upright orientation but [LZL16] unexpectedly rotates it to a non-upright direction. In contrast, our model can well capture and predict the correct upright orientations of these models, regardless of their original orientations being near or far away from the ground truths.

We also illustrate more results obtained by our method in Figure 7 to present the performance on the three datasets with var-

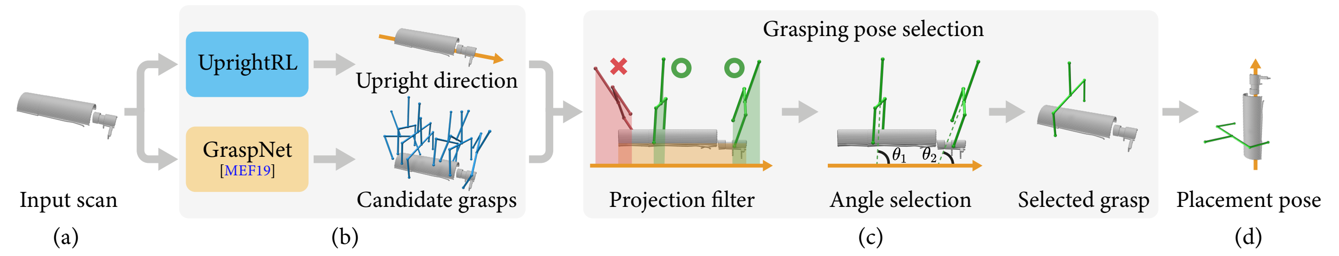

Figure 8: *Upright orientation guided grasp selection and placement pose determination method. For the given input scan represented by partial point cloud, we apply our UprightRL model to estimate the upright orientation and the method of [MEF19] to predict a set of candidate grasping poses. Then we filter out unreasonable grasping poses by comparing their projections and the object's projection along the upright direction, and further select the optimal grasping pose based on the grasping angle relative to the upright direction. The final placement pose is determined by rotating the object so that it's upright direction is along z-axis.*

Table 3: *Comparison between different angle step update strategies on the single scan dataset.*

Strategy	$\leq 15^\circ$	$\leq 10^\circ$	$\leq 5^\circ$
Decreased	0.910	0.866	0.623
Multiple	0.927	0.891	0.754
Fixed	**0.927**	**0.902**	**0.758**

ious completeness. As seen, despite of the various completeness and random rotation of the input point cloud, our model predicts accurate upright orientations of them. Taking the Cabinet (column 2) as an example, regardless of the input shape orientations including already in, nearly in or far away from upright orientations, our method is able to consistently predict its upright orientation and further preserve, fine-tune or correct the pose of the input shape, respectively.

4.3. Ablation study of angle step

In our experiments, a rotation with a fixed angle step is used when the point cloud is updated at each time step. To better understand the impacts of angle step, we conducted the following ablation study with three different angle step update strategies for our UprightRL model as follows:

- Decreased: We gradually decreased angle step during training. Specifically, we set $\omega = 8^\circ$ in steps $1 \sim 6$, $\omega = 4^\circ$ in steps $7 \sim 12$, and $\omega = 2^\circ$ in steps $13 \sim 20$.
- Multiple: Instead of fixed angle step, the angle step ω is selected from a candidate set $\Omega = \{2^\circ, 4^\circ, 8^\circ\}$.
- Fixed: We use a fix angle step where $\omega = 4^\circ$.

We train and test the UprightRL model with these strategies on the single scan dataset. As shown in Table 3, the fixed strategy produces the highest accuracy, while the performance of multiple angle steps is rather close to the fixed one. As revealed by the table, choosing the angle step by network itself could be more reasonable than decreasing the angle step by ourselves.

Table 4: *Testing accuracy under [LZL16] dataset in ModelNet for the shared categories using our UprightRL model trained on Complete point cloud dataset.*

Category	Airplane	Car	Chair	Table
Accuracy	0.920	0.980	0.920	0.960

4.4. Generalization

To better evaluate the generalization of our method, we further conducted an additional experiment where we trained the network on Complete point cloud dataset generated from ShapeNet and tested it on ModelNet [LZL16] for the shared categories. As shown in Table 4, the accuracy of angle errors within 15° is above 0.9 over different categories and close to the result tested by the network trained on ModelNet, demonstrating good generalization of our UprightRL model on synthetic data.

5. UprightRL for Robot Grasping-and-Placing

In this section, we show that upright orientation estimation of 3D shapes can be applied to the robot grasping-and-placing task. Our key insight is that for certain scenarios of robot grasping and placing, it not only requires robots to grasp objects successfully, but also to place objects steadily. By involving upright orientation estimation and using it to filter out unreasonable grasping poses and select the optimal grasping pose, our experiments demonstrate that UprightRL can significantly improve the placement efficiency.

5.1. Algorithm

Unlike previous methods that tackle the grasping problem and placing problem separately, we propose to optimize them jointly: selecting a suitable grasping pose for better placing. Intuitively, it is generally easier and more steady to place an object with its upright orientation, which inspires us to consider orientation estimation during grasping and placement. Towards this end, we propose a novel grasp selection and placement pose determination method, as shown in Figure 8. To simplify visualization, we use a gripper skeleton to represent the gripper.

Grasping pose prediction. We apply the method of [MEF19] for grasping pose prediction as it performs well in both simulated and real robot experiment environments. For each input point cloud, it output a set of 6-DOF grasping poses with associated confidence scores. To ensure these grasps are executable and the gripper will not collide with table when performing the grasping, we further remove the grasping poses where the angle between its grasp direction and the negative z-axis is larger than 45°, which results in a set of candidate grasping poses as shown in Figure 8 (b) in blue.

Grasping pose selection. With associated confidence scores, the most straightforward way for pose selection is to use the grasping pose with highest confidence. However, to ensure the stability of the final placement, we use the upright direction to guide the grasping pose selection, which consists of two steps: projection filter and angle selection, as shown in Figure 8 (c).

The projection filter is used to filter out the grasping poses that are unable to place the object in its upright orientation. For example, if we grasp the object from its bottom, then the gripper will collide with the table first before placing the object. Specifically, We compare the projections of gripper key points and input scan along the predicted upright direction. If any part of the projection region of the gripper lies outside the projection region of the input scan along the negative upright direction, we consider the corresponding grasping pose invalid. Figure 8 (c) shows one invalid grasping pose in red and two valid grasping poses in green.

Among all those valid grasping poses, we further select the one with grasping direction most perpendicular to the upright direction, which leads to more stable placement in the end. We compute the cosine of angle between the upright direction and the plane of the gripper skeleton, and select the grasping pose with lowest value.

Placement pose determination. Once the object is grasped, we need to further determine the final gripper pose to place the object in a predefined position. Since the object will be in a more stable state when put in its upright orientation, we rotate the object and the gripper together to get the final placement pose so that the upright direction is along the z-axis in the end. Figure 8 (d) shows one example of the final placement pose.

5.2. Experiments

We now introduce the details of the experiment setting and evaluate the effectiveness of our UprightRL model when applied on the robot grasping-and-placing task.

Data preparation. We select 10 bottles from ShapeNet [CFG*15] to test our method. Each bottle is rotated around the z-axis by a angle $\theta \in \{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ, 180^\circ\}$ after putting on the table to get different input configurations, which results in 70 testing cases in total.

Placement quality. We use PyBullet [CB21] as our robot simulation environment, and control the robot motion based on Samuel Buss Inverse Kinematics library [BK05]. Since the object may still be shaking after placing on the table for a long time in the simulator, we consider the placement is stable after the simulator runs for 200 more timesteps and then check the angle between the final upright direction of the object and the z-axis. A placement is considered to be successful if the angle is less than 15°.

Table 5: *Placement success rate comparison in different settings for the robot grasping-and-placing task.*

	Grasp w/o upright	Grasp with upright
Place w/o upright	0.143	0.257
Place with upright	0.300	**0.486**

Experimental results. Table 5 shows the placement success rate comparison in different settings for the robot grasping-and-placing task. For grasping pose selection, we consider two options: one uses our upright-based selection method and the other simply picks the pose with highest confidence. For final placement pose, we also consider two options: one uses our upright-based determination method and the other simply uses the horizontal gripper pose. As seen in Table 5, the highest success rate is achieved when using the upright direction to guide both grasping pose selection and placement pose determination, which indicates the effectiveness of our UprightRL model applied in the robot grasping-and-placing task. We also show some visual examples of grasping-and-placing sequences in Figure 9. Compared with [FCDS08], our proposed UprightRL can handle partial observation instead of a full 3D object input, which is a more practical setting for robot manipulation.

6. Conclusion and Future Work

We present a reinforcement learning based method to solve the problem of 3D shape upright orientation estimation. We model this task as a sequential decision-making process with customized action space as well as reward function. Our UprightRL model works well for various input forms, ranging from complete and partial point cloud to single scan data. It outperforms previous state-of-the-art method and other alternative supervised learning based baseline with a large margin. We also demonstrate its application on robot grasping-and-placing task and show promising improvements.

Limitations and future work. Although our method shows a great advantage on the task of 3D shape upright orientation estimation compared with existing methods, there exist some limitations for further investigation. First of all, the number of object categories in current datasets is not sufficiently large, and we would like to experiment our method with more diverse object categories to explore its generalization. Secondly, compared with previous methods, it takes much longer time to train UprightRL due to its sequential decision-making process, and it would be helpful to accelerate the training. Last but not the least, currently we only show the effectiveness of our method when applied to the robot grasping-and-placing task in a simulated environment, and it would be interesting to experiments in real environment in the future. Different from the clean point cloud data obtained from simulator, there could be other factors like segmentation and noise that can low down the performance for data in real scenarios. How to improve the robustness in real scenarios would be a direction for our future work.

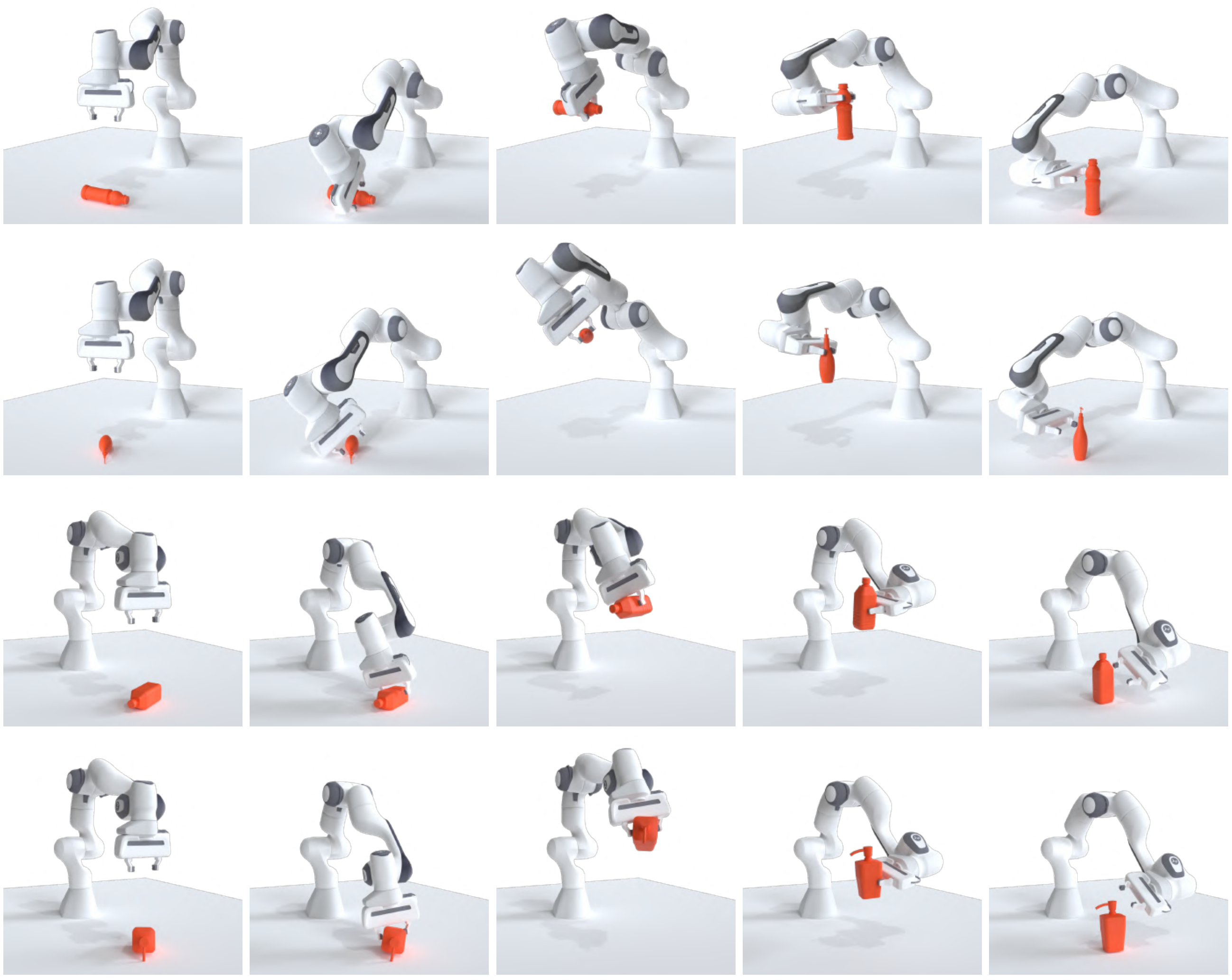

Figure 9: *Example results of the robot grasping-and-placing experiment. Each row shows the sequence from grasping to placing an object.*

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was supported by NSFC (61872250), GD Natural Science Foundation (2021B1515020085), GD Talent Program (2019JC05X328), DEGP Key Project (2018KZDXM058), Shenzhen Science and Technology Program (RCJC20200714114435012), National Engineering Laboratory for Big Data System Computing Technology, and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

References

[ADBB17] ARULKUMARAN, K., DEISENROTH, M. P., BRUNDAGE, M., and BHARATH, A. "Deep Reinforcement Learning: A Brief Survey". *IEEE Signal Processing Magazine* 34.6 (2017), 26–38. DOI: 10.1109/MSP.2017.2743240 2.

[BK05] BUSS, SAMUEL R and KIM, JIN-SU. "Selectively damped least squares for inverse kinematics". *Journal of Graphics tools* 10.3 (2005), 37–49 9.

[BPL*17] BELLO, IRWAN, PHAM, HIEU, LE, QUOC V., et al. "Neural Combinatorial Optimization with Reinforcement Learning". (2017). URL: https://openreview.net/forum?id=Bk9mxlSFx 3.

[BRHS14] BOHG, J., ROMERO, J., HERZOG, A., and SCHAAL, S. "Robot arm pose estimation through pixel-wise part classification". *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, 3143–3150. DOI: 10.1109/ICRA.2014.6907311 5.

[CB21] COUMANS, ERWIN and BAI, YUNFEI. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021 9.

[CFG*15] CHANG, ANGEL X., FUNKHOUSER, THOMAS A., GUIBAS, LEONIDAS J., et al. "ShapeNet: An Information-Rich 3D Model Repository". *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012. URL: http://arxiv.org/abs/1512.03012 9.

[FCDS08] Fu, F., Cohen-or, D., Dror, G., and Sheffer, A. "Upright orientation of man-made objects". *ACM Trans. Graphics* 27.3 (2008), 1–7. doi: 10.1145/1360612.1360641 2, 9.

[HZY*17] Hu, Haoyuan, Zhang, Xiaodong, Yan, Xiaowei, et al. "Solving a new 3d bin packing problem with deep reinforcement learning method". *arXiv preprint arXiv:1708.05930* (2017) 3.

[JWL12] Jin, Y., Wu, Q., and Liu, L. "Unsupervised upright orientation of man-made models". *Graph. Model.* 74.4 (2012), 99–108. doi: 10.1016/j.gmod.2012.03.007 2.

[KB14] Kingma, Diederik P and Ba, Jimmy. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 5.

[LWZH18] Li, D., Wu, H., Zhang, J., and Huang, K. "A2-RL: Aesthetics Aware Reinforcement Learning for Image Cropping". *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*. 2018, 8193–8201. doi: 10.1109/CVPR.2018.00855 2, 3.

[LZL16] Liu, Z., Zhang, J., and Liu, L. "Upright orientation of 3D shapes with convolutional networks". *Graph. Model.* 27.85 (2016), 22–29. doi: 10.1016/j.gmod.2016.03.001 2, 5–8.

[MBM*16] Mnih, V., Badia, A. P., Mirza, M., et al. "Asynchronous Methods for Deep Reinforcement Learning". *International conference on machine learning*. Vol. 48. 2016, 1928–1937 2, 4.

[MEF19] Mousavian, A., Eppner, C., and Fox, D. "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation". *2019 IEEE/CVF International Conference on Computer Vision, ICCV*. IEEE, 2019, 2901–2910. doi: 10.1109/ICCV.2019.00299 8, 9.

[PGM*19] Paszke, Adam, Gross, Sam, Massa, Francisco, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems 32*. Ed. by Wallach, H., Larochelle, H., Beygelzimer, A., et al. Curran Associates, Inc., 2019, 8024–8035. url: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 5.

[PHBD21] Park, Sohee, Hoai, Minh, Bhattacharya, Arani, and Das, Samir R. "Adaptive streaming of 360-degree videos with reinforcement learning". *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, 1839–1848 3.

[QSMG17] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (17), 652–660 4, 5.

[QYSG17] Qi, Charles R, Yi, Li, Su, Hao, and Guibas, Leonidas J. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". *arXiv preprint arXiv:1706.02413* (2017) 4.

[TKR*19] Tchapmi, Lyne P, Kosaraju, Vineet, Rezatofighi, S. Hamid, et al. "TopNet: Structural Point Cloud Decoder". *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 5.

[VFJ15] Vinyals, Oriol, Fortunato, Meire, and Jaitly, Navdeep. "Pointer Networks". (2015). Ed. by Cortes, C., Lawrence, N. D., Lee, D. D., et al., 2692–2700. url: http://papers.nips.cc/paper/5866-pointer-networks.pdf 3.

[WLL14] Wang, W., Liu, X., and Liu, L. "Orientationof 3D shapes via tensor rank minimization". *Graph. Model.* 27.7 (2014), 2469–2477. doi: 10.1007/s12206-014-0604-6 2.

[WP91] Williams, R. J. and Peng, J. "Function optimization using connectionist reinforcement learning algorithms". *Connection Science* 3.3 (1991), 241–268 4.

[YKH*18] Yuan, W., Khot, T., Held, D., et al. "PCN: Point Completion Network". *2018 International Conference on 3D Vision (3DV)*. 2018, 728–737. doi: 10.1109/3DV.2018.00088 5.

[YSGG17] Yi, L., Su, H., Guo, X., and Guibas, L. J. "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation". *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2017, 6584–6592. doi: 10.1109/CVPR.2017.697 5.

[ZYD21] Zhang, Youshan, Ye, Hui, and Davison, Brian D. "Adversarial Reinforcement Learning for Unsupervised Domain Adaptation". *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2021, 635–644 3.