PPN-Pack: Placement Proposal Network for Efficient Robotic Bin Packing

Jia-Hui Pan, Xiaojie Gao, Ka-Hei Hui, Shize Zhu, Yun-Hui Liu, *Fellow, IEEE*, Pheng-Ann Heng, *Senior Member, IEEE*, and Chi-Wing Fu

***Abstract*—Robotic bin packing is a challenging task, requiring compactly packing objects in a container and also efficiently performing the computation, such that the robot arm need not wait too long before taking action. In this work, we introduce PPN-Pack, a novel learning-based approach to improve the efficiency of packing general objects. Our key idea is to learn to predict good placement locations for compact object packing to prune the search space and reduce packing computation. Specifically, we formulate the learning of placement proposals as a ranking task and construct a ranking loss based on the Hinge loss to rank the potentially compact placements. To enhance the placement learning, we further design a multi-resolution cross-correlation module to better exploit the placement compactness between the container and objects. We perform extensive experiments on 2,000 packing cases. By equipping PPN-Pack to prune the proposals for the upper-bound packing heuristic, we can yield a remarkable 76% reduction in computation time, while achieving comparable packing compactness. Compared with the other packing heuristics, PPN-Pack reduces at least 47% computation time, while producing more compact packings.**

***Index Terms*—Manipulation planning, visual learning, integrated planning and learning.**

I. Introduction

BIN packing involves compactly arranging objects within a container by iteratively selecting an object and determining its feasible and compact placement location. It is known to be NP-hard [1], [2] given the extensive search space of object choices and placement options. General bin packing tasks [3], [4], [5] mainly consider non-collision placements for objects to improve space utilization.

Robotic bin packing in industrial settings, like auto warehouses and unmanned stores, is more intricate, since gravity and robot-arm feasibility need to be considered. Packing regularly-shaped objects using robots such as boxes has already drawn much research attention recently [1], [2], [6], [7], [8]. The task becomes even more challenging when we aim to handle general objects of non-regular shapes and varying sizes. It is very hard to simultaneously achieve high computational efficiency and high packing compactness, due to the extensive search space and high geometric complexity.

To meet the need for computational efficiency, various heuristics are proposed [9], [10], [11]. Overall, they propose various forms of simple-to-compute objective functions to efficiently evaluate object placements. At each packing step, one or more candidate objects are considered. First, the object placements (i.e., orientations and placement locations) are discretized. Then, a grid search is performed to exhaustively compute a heuristic value for every object and placement to determine the optimal object and placement. However, the search space is still large which has hindered the computational efficiency.

To overcome this limitation, we present PPN-Pack, a novel approach with a placement proposal network, namely PPN, to swiftly locate potentially compact placement proposals for a given candidate object by predicting the quality of placements, such that the search on non-compact placements can be largely pruned. The reasons for adopting a neural network are as follows. First, neural networks can learn intricate functions effectively. In our case, the PPN can learn the quality of different placements based on a ground-truth heuristic function. Second, neural networks can be fast in inference by means of parallel computation. The inference time for PPN is only 0.02 seconds. Also, neural networks can generalize to unseen data. The PPN can handle novel packing scenes to speed up the packing computation. As Fig. 1 shows, our PPN-Pack leads in the best time efficiency compared with the existing heuristics, while producing highly compact packing.

In detail, the placement proposal learning is formulated as a supervised ranking task. We train the PPN by adopting a ranking loss to prioritize optimal placements (including object orientation and location) for each candidate object based on the geometrical condition of the container. To compare placement locations at a certain object orientation, the location loss is

Manuscript received 5 December 2023; accepted 22 March 2024. Date of publication 5 April 2024; date of current version 19 April 2024. This letter was recommended for publication by Associate Editor A. Sintov and Editor J. Borràs Sol upon evaluation of the reviewers' comments. This work was supported in part by the Shenzhen Portion of Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under Grant HZQB-KCZYB-20200089, in part by the InnoHK of the Government of Hong Kong via the Hong Kong Centre for Logistics Robotics, and in part by the CUHK T Stone Robotics Institute. *(Corresponding author: Chi-Wing Fu.)*

Jia-Hui Pan, Xiaojie Gao, Ka-Hei Hui, Pheng-Ann Heng, and Chi-Wing Fu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: jhpan21@cse.cuhk.edu.hk; xjgao@cse.cuhk.edu.hk; khhui@cse.cuhk.edu.hk; pheng@cse.cuhk.edu.hk; cwfu@cse.cuhk.edu.hk).

Shize Zhu is with the Hong Kong Centre for Logistics Robotics, Hong Kong (e-mail: szzhu@hkclr.hk).

Yun-Hui Liu is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: yhliu@mae.cuhk.edu.hk).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3385612, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3385612

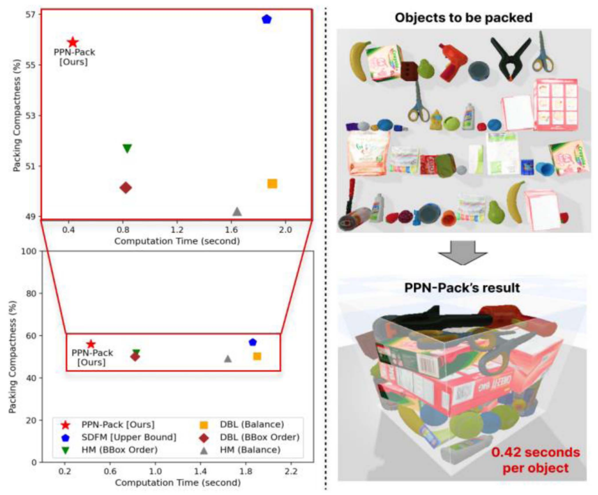


Fig. 1. Left: our PPN-Pack can greatly reduce the computation time, while maximizing the packing compactness, compared with the existing packing heuristics. SDFM, DBL, HM, Balance, and BBox Order denote the SDF-Minimization [11], the Deepest Bottom-Left [10], the Heightmap-Minimization [9], the size-balancing term [11] for object ordering, and bounding-box volume decreasing object order, respectively. For more details, please refer to Section IV-B. Right: an example packing result produced by PPN-Pack. On average, it takes only 0.42 seconds to find an object placement.

introduced to rank the optimal location higher than the others. Also, a global loss is designed to prioritize the optimal placement across different object orientations. Further, to better assess the compactness of placements, the multi-resolution cross-correlation module is devised to model the geometric correlation between an object and a local region around the object in the container. This is achieved by using the object's geometric feature map as a kernel and convolving it with the container's geometric feature map at varied resolutions. The contributions of this work are three-folded:

i) We introduce PPN-Pack, a new approach to improve the packing efficiency for general objects by designing the PPN to learn to predict good placement proposals;
ii) We formulate the placement proposal learning as a ranking task and formulate a ranking loss based on the Hinge loss to prioritize the optimal placements;
iii) To facilitate placement learning, we design the multi-resolution cross-correlation module to exploit the compactness of object placement in the container.

Experiments on 2,000 packing sequences show that by using PPN-Pack, we can prune a large portion of placement locations for the upper-bound packing heuristic to work on, achieving a huge reduction in packing computation time by 76%. Compared with other heuristics, PPN-Pack can reduce the computation time by at least 47%, while achieving much more compact packing in the results; see Fig. 1.

II. Related Works

Bin packing has been extensively studied in computer graphics [3], [4], [5]. However, general bin-packing methods are inadequate for robotics applications because they overlook gravity and robot-arm feasibility, and are not optimized in computation time. Many robotic bin packing methods are proposed, yet most of them focus on regularly-shaped objects [1], [2], [6], [7], [8]. In more complex industrial scenes, handling irregularly shaped objects of varying sizes presents challenges for computational efficiency and packing compactness.

Many attempts have been made to improve the computational efficiency of bin packing. Some methods try to speed up the optimization of the object selection at each packing step via genetic algorithm [10], [12], [13], tabu search [14], simulated annealing [15], or integer linear programming [16], [17]. They attempt to optimize the object selection within a subset instead of searching over all combinations of object selections. However, their computational efficiency is still very limited, since they need to extensively try many packing sequences of different object choices.

Some other works simplify the search by pre-defining a sequence of object choices and optimizing the object placement by only looking ahead to fewer packing steps. They propose data structures such as Markov decision tree [18], [19] and packing configuration tree [20] for the placement search. However, as the combinations of placement options still yield a large search space, even for 5 to 10 packing steps, analyzing all possible future steps is still too time-consuming.

Very recently, reinforcement learning (RL) is introduced to robotic bin packing [1], [2], [21], [22], and in particular, [1], [22] attempt to handle non-regular objects. RL methods learn to optimize the multi-step packing results. However, they still require sampling many trials of the same packing case for training and have limited robustness towards unseen objects.

To meet the need for fast computation for supporting robotic bin packing, some heuristics [9], [10] are proposed to quickly determine the best object choice and placement options with an objective function. For example, the Deepest Bottom-Left (DBL) [10] prioritizes the placements close to the Deepest Bottom-Left corner; the Heightmap-Minimization (HM) [9] recommends the placements that minimize heightmap increments; and the SDF-Minimization [11] encourages compact packing by minimizing the signed distance field (SDF) values, achieving state-of-the-art packing compactness. These methods can be applied to real-world bin-packing sequences on the fly without sampling multiple packing sequences in advance, and they are robust to unseen novel shapes. As a result, they have long been a favorite in industrial applications. However, they still require extensive evaluations of all object placement locations in the container for every packing step. PPN-Pack addresses these challenges with a novel approach that learns to quickly identify good-quality placement proposals. Hence, we can prune the search space and make efficient the placement evaluation, while maintaining similar packing compactness. It also generalizes well to test packing sequences and unseen objects. Please refer to Section IV for quantitative evaluations that show PPN-Pack's superiority.

III. Method

A. Problem Definition

Given a container and a sequence of objects that arrive in order, a robotic bin packing procedure involves multiple packing steps, in which an object is chosen and packed at a target

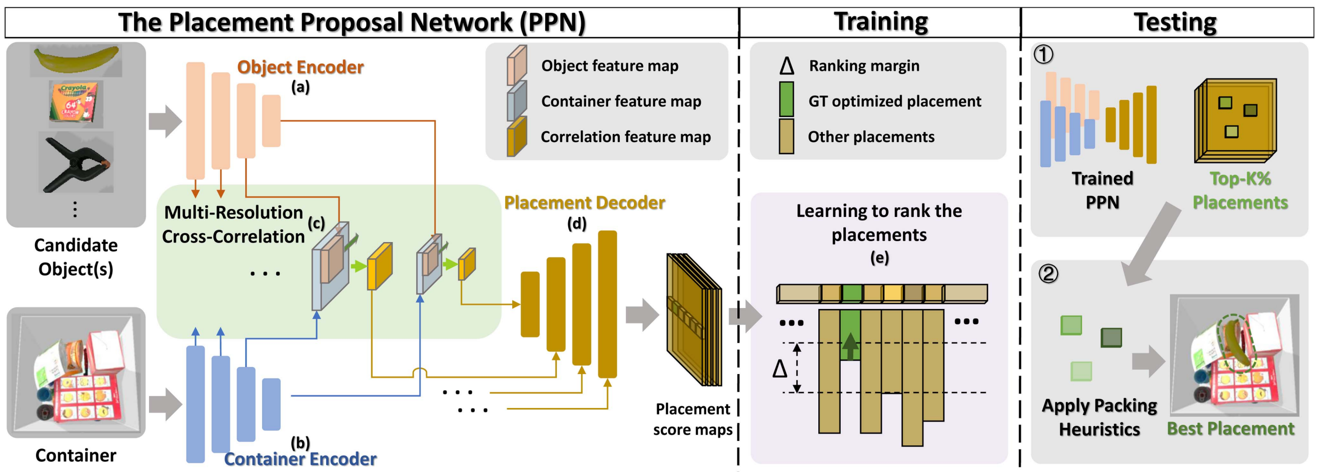


Fig. 2. Illustration of the placement proposal network, the network training and testing. Given multiple candidate objects and a container, our PPN predicts a placement score map. We feed a pair of top-down and bottom-up heightmaps for each candidate object to the object encoder (a) and feed the top-down heightmap of the container to the container encoder (b) to obtain the object and the container feature maps. Then, we learn the geometric correlation between each candidate object and the container via the multi-resolution cross-correlation module (c) and generate placement score maps using the placement decoder (d). We train the PPN to prioritize the optimal placement by ranking it better than the others (e) using a ranking loss. In inference, our PPN can quickly generate the top-$K\%$ good placement proposals, reducing the search space and speeding up the packing heuristics computation(①, ②). Best viewed in color.

placement inside the container in each packing step. Following the setting of [11], the object is chosen from a buffer that contains the first B objects yet to be packed. The packing procedure ends when all objects are successfully packed, or there is no feasible location to place any object in the buffer.

In each packing step, a packing heuristic [9], [10], [11], whose objective value is often the lower the better, is used to evaluate the compactness of each object and placement. So, the optimal object b^* and placement p^* are given by

$$b^*, p^* = \arg\min_{b,p} F(b,p) \quad \forall p \in \mathcal{P}, \forall b \in \{1,2,3,\ldots,B\}, \tag{1}$$

where b is the index of a candidate object from the buffer; $\mathcal{P}$ is the set of all candidate placements, with each placement p described by an object orientation r and a horizontal location (x, y) to insert the object towards the deepest limit, i.e., $p = (r, x, y)$; $F(\cdot)$ is the packing heuristic, avoiding infeasible choices by yielding ∞ as their outputs.

B. Our Packing Framework

While packing heuristics are fast, the requirement to compare all objects and placements hinders their computational efficiency. Thus, we present the PPN-Pack, which predicts the best-valued placement proposals for each object, to reduce the number of placements that need to be evaluated.

Our PPN-Pack achieves this goal by training a placement proposal network (PPN) to rank the optimal placements higher than others. Fig. 2 shows the framework of our method. In each packing step, we first encode the geometry of the container and each candidate object at varied resolutions. Subsequently, we formulate a multi-resolution cross-correlation module to capture the geometric relations between each candidate object and the distinct local regions of the container where the object may be placed. Lastly, we decode the correlation features to form the placement score maps, which indicate the compactness of packing a certain object b at a certain placement p. We formulate the placement learning with a ranking-learning scheme to prioritize the optimal placements labelled by a packing heuristic. Thus, in inference, we can rapidly identify the optimal object and placement, (i.e., (b^*, p^*)) by evaluating the top-$K\%$ placements instead of searching all over the entire container.

In the following, we will first introduce the placement proposal network for learning the placement score maps (Section III-C). Then, we introduce the training loss to rank the optimal placements (Section III-D). Finally, we present how PPN is used to accelerate packing in inference (Section III-E).

C. The Placement Proposal Network (PPN)

We follow [9] to generate a pair of top-down and bottom-up heightmaps for each candidate object, and a top-down heightmap for the container, by casting rays to a subject's 3D model to measure the height from its top (or bottom) surface to the opposite plane of its 3D bounding box. Given these heightmaps as input, the PPN learns a placement score map $\mathbf{M}$ for each candidate object. Each element in $\mathbf{M}$ indicates the compactness of packing the object at orientation r and horizontal location (x, y). See Fig. 3 for an illustration.

The encoders: Given the input heightmaps, we employ the object and the container encoders to predict the geometric feature maps for the objects and the container, respectively. Both encoders contain four encoder blocks, each containing two 2D convolution layers and a max-pooling layer. They predict the geometric feature maps and gradually down-sample them to varied resolutions. In the following, we indicate the object feature map and the container feature map at each resolution as $\mathbf{K}^i$ and $\mathbf{H}^i$, respectively, with $i = 1, 2, 3$ and 4. As the value of i increases, the resolution decreases.

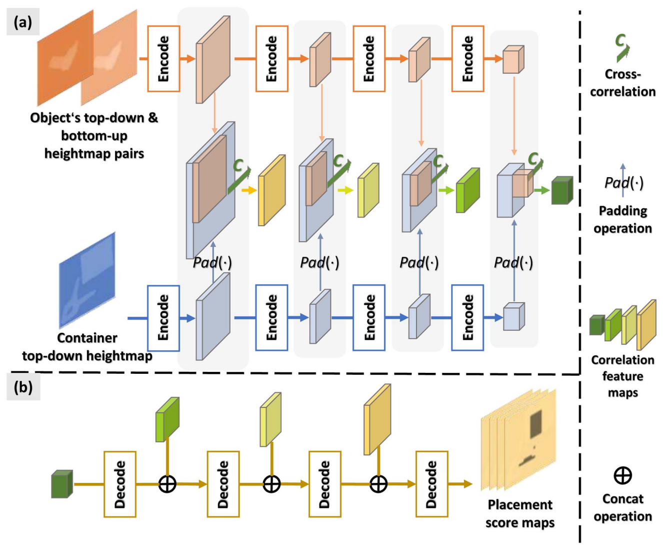


Fig. 3. (a) Encoding and multi-resolution cross-correlation. We obtain multi-resolution feature maps for a candidate object (in orange) and the container (in blue) via separate encoders and then conduct cross-correlation learning at varied resolutions. This entails treating the object feature map as a kernel for convolution with the container's feature map, producing multi-resolution correlation feature maps. (b) The decoding process. The multi-resolution correlation feature maps are concatenated and up-sampled by four decoder blocks to form the placement score map. Best viewed in color.

The multi-resolution cross-correlation module: Then we model the geometric relations between a candidate object and the container, which can be further used for analyzing the packing compactness. More specifically, to analyze a certain object placement involves modeling the geometry of the entire object and that of the local region of the container where the object is placed. We achieve this by adopting cross-correlation [23], [24], [25], which uses the object's feature map as a kernel to convolve with the search area. In our context, we employ cross-correlation learning on the object feature map $\mathbf{K}^i$ and the container feature map $\mathbf{H}^i$ to yield cross-correlation feature map $\mathbf{C}^i$ at each resolution i, which is written as

$$\mathbf{C}^i = Pad(\mathbf{H}^i) * \mathbf{K}^i, i = 1, 2, 3, 4, \tag{2}$$

where $*$ represents a 2D convolution operation; $\mathbf{C}$ is the resulting cross-correlation feature map; $Pad(\cdot)$ represents a padding operation such that $\mathbf{C}^i$ maintains the same size as $\mathbf{K}^i$ and $\mathbf{H}^i$. Each element in $\mathbf{C}^i$ represents the geometric relation between a certain candidate object and a local container region, which is further leveraged to learn the packing compactness of the candidate object when placed at different placements p.

The placement decoder: After acquiring the correlation feature maps $\mathbf{C}^i$ at varied resolutions i, we feed them to the placement decoder (refer to Fig. 3(b)) to generate the placement score map $\mathbf{M}$. The placement decoder consists of four blocks, each comprising a batch-normalization layer and a 2D de-convolution layer to up-sample the $\mathbf{C}^i$. The resulting feature map is concatenated with the $\mathbf{C}^{i-1}$ of a higher resolution and fed to the next decoder block. The output of the decoder is the placement map $\mathbf{M}$ associated with the input candidate object. It consists of two spatial dimensions, which have the same size as the input container heightmap, and an additional dimension to represent different object orientations.

D. Ranking the Optimal Placements

After acquiring the placement map $\mathbf{M}$ for each candidate object, we train the PPN to prioritize the optimal placements. Following the definition of the optimal object and placements for a packing step in Section III-A, we define further the optimal placement p^* for each candidate object b as

$$p^* = (r^*, x^*, y^*) = \arg\min_{r,x,y} F(r, x, y; b), \tag{3}$$

where r is the orientation of the object; x and y denote the horizontal position; and $F(\cdot)$ is the packing heuristic function (lower is better). Similarly, for each candidate object, we can define the best placement location given a certain object orientation as $(x^*, y^*; r) = \arg\min_{x,y} F(x, y; b, r)$. The labels of the optimal placements are employed to train our PPN. In this work, we use the SDF-Minimization heuristic [11], as this heuristic achieves state-of-the-art packing compactness.

The goal of our placement proposal learning is to train the PPN to distinguish the optimal placement and the other placements, such that we can then obtain good placement proposals by taking a small portion of best-valued placements. To be specific, we want PPN's resulting placement score map $\mathbf{M}$ to satisfy $\mathbf{M}(p^*) < \mathbf{M}(p)$ for the optimal placement p^* and other placements p. Therefore, we formulate placement proposal learning as a ranking task. We develop our ranking losses based on the Hinge loss, which has succeeded in various ranking applications such as skill assessment in human actions [26], [27] and image-based age estimation [28], [29], but has not been exploited in object bin packing.

Considering that comparing the placement locations under a fixed orientation is an easier task than comparing different locations and orientations simultaneously, we first introduce the location loss to distinguish the optimal placement location at each orientation, which is written as

$$L^r_{loc} = \sum_{x,y} \max(0, \Delta_{loc} - (\mathbf{M}(x, y; r) - \mathbf{M}(x^*, y^*; r)), \tag{4}$$

where Δ_{loc} is the ranking margin, which encourages that for any orientation r, the output score for non-optimal locations (x, y) is worse than that of the optimal location (x^*, y^*) by at least Δ_{loc}.

Further, we design the global ranking loss to prioritize the optimal placement across varied orientations:

$$L_{glo} = \sum_{r,x,y} \max(0, \Delta_{glo} - (\mathbf{M}(r, x, y) - \mathbf{M}(r^*, x^*, y^*)), \tag{5}$$

where Δ_{glo} is the ranking margin, encouraging the optimal placement $p^* = (r^*, x^*, y^*)$ to score better than others by at least Δ_{glo}.

We integrate (4) and (5) to form the total ranking loss:

$$L_{rank} = \frac{1}{|\mathcal{P}|}\left(\sum_r L^r_{loc} + L_{glo}\right), \tag{6}$$

Algorithm 1: Packing Planning in Testing.

```
Inputs: the top-down heightmap of the container $I_C$ and
  the top-down and bottom-up heightmap pair $I_O^b$ of each
  candidate object $b$; the packing heuristic function $F(\cdot)$;
  the portion of placement proposals towards all placements
  $K\%$.
Outputs: the optimal objective value $h^*$ and the
  corresponding optimal object and placement choice $\mathcal{T}$.
 1: $h^* \leftarrow \infty, \mathcal{T} \leftarrow \emptyset, \mathcal{P} \leftarrow \{p \mid \forall p\}$
 2: for each candidate object $b$
 3:     $\mathbf{M} \leftarrow PPN(I_C, I_O^b)$
 4:     $\hat{\mathcal{P}} \leftarrow \{p \mid \forall p \text{ s.t. } \mathbf{M}(p) \leq TopK(\mathbf{M}, |\mathcal{P}| \cdot K\%)\}$
 5:     for all $\hat{p} \in \hat{\mathcal{P}}$ do
 6:         if IsValid[1] $(\hat{p})$ and $F(b, \hat{p}) < h^*$. then
 7:             $h^* \leftarrow F(b, \hat{p})$
 8:             $\mathcal{T} \leftarrow (b, \hat{p})$;
 9:         end if
10:     end for
11: end for
12: return $h^*$, $\mathcal{T}$
```

where $\mathcal{P}$ is the set of all candidate placements, including different orientations and locations. We use $\Delta_{glo} = 2 * \Delta_{loc}$ to encourage the global optimal placement to stand out from all the other placements including the locally optimal ones (r, x^*, y^*) for each orientation. Note that we can handle multiple optimal placements by ranking all optimal placements higher than the non-optimal ones.

E. Test-Time Packing Acceleration

In inference, our trained PPN can be utilized to accelerate the packing computation for each packing step. The algorithm for such a procedure is shown in Algorithm 1. For each candidate object b, we obtain the placement score map $\mathbf{M}$ using the trained PPN, and then retrieve the top-$K\%$ best-valued placements to form the proposals $\hat{\mathcal{P}}$ (see Lines 3-4). Then we examine only $\hat{\mathcal{P}}$ instead of all placements ($\mathcal{P}$) to find the optimal placement (see Lines 5-8).

Considering discrete placement locations of $N \times N$, and R discrete object orientations, the heuristic method's computational complexity is $\mathcal{O}(1 \cdot R \cdot N^2)$. Conversely, PPN-Pack has a complexity of $\mathcal{O}(K\% \cdot R \cdot N^2 + G)$ as it only examines the top-$K\%$ placement proposals with an additional network prediction time of G. In our case, the G is only 0.02 seconds, constituting only 1% of $\mathcal{O}(1 \cdot R \cdot N^2)$ (1.86 seconds). Despite having the same worst-case quadratic scaling, PPN-Pack can offer a faster solution through a trade-off between packing compactness and time efficiency. Setting K close to 0 ensures faster computation but may lead to sub-optimal placements, while setting K close to 100 ensures choosing the optimal placements with increased computation time. In our experiments, setting K=5 yielded a remarkable 76% reduction in computation time. Note that the time reduction is not precisely (100-K)% since some pruned placements are infeasible with faster-to-compute heuristic values ∞ (See Section III-A).

[1]IsValid($\cdot$) is **True** if the object can be packed inside the container and passes the stability test in [11].

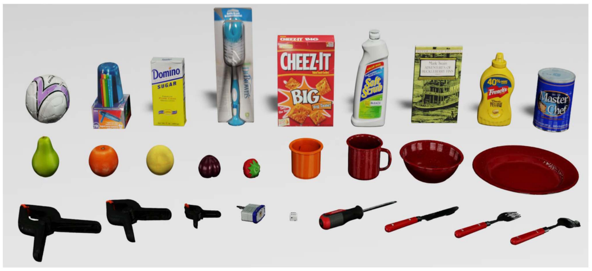


Fig. 4. Some object examples from the dataset used in our experiments. The dataset covers a diversity of varied object shapes and sizes.

IV. Experiments

A. Implementation Details

Datasets: Following the setting of [11], we perform experiments on a set of 96 real-world objects: 71 from the YCB dataset [30] and 25 from the Rutgers APC RGB-D dataset [31]. Fig. 4 shows some of these objects. In detail, the 3D mesh of each object is pre-processed to be watertight using the Manifold algorithm [32] and then decomposed into convex components using the V-HACD algorithm [33] for building the collision models employed in physical simulation.

Sequential packing: To evaluate the sequential-packing performance of different methods towards the limit of the container, we sample 2,000 packing sequences, each with 80 objects randomly selected from the dataset. As described in Section III-A and Algorithm 1, our method iteratively chooses the optimal object from a buffer with the first B objects yet to be packed and finds the optimal placement among the top-$K\%$ best-valued placements as the placement proposals. We set the buffer size $B = 5$ as in [11] and retrieved the top-5% placements as the placement proposals ($K = 5$).

Network training: To prepare for the training and testing of the placement proposal network, we additionally sample 1,000 packing sequences following the above setting. Then, we apply the SDF-Minimization heuristic [11] to annotate the ground-truth optimal placement in each packing step. The sampled sequences are split by 9:1 to form a training and a testing set, where the testing set is used to analyze PPN's performance in Section IV-D. We used the AdamW optimizer with a learning rate of 10^{-4} to train our network for 25 epochs. The training process is performed on an NVIDIA GeForce RTX 2080 for 3 GPU hours. The PPN's inference time is only 0.02 seconds.

Physical simulation: All packing experiments are performed using the PyBullet [34] simulator. We set up a top-down depth camera to acquire the heightmap of the container, and a pair of top-down and bottom-up cameras to capture the heightmap pair of each object. We follow the existing works [9], [11] to use a 32 cm $\times$ 32 cm $\times$ 30 cm container. All the heightmaps are discretized by 1 cm in the x and the y dimensions and

TABLE I
COMPARING THE AVERAGE PERFORMANCE OF DIFFERENT PACKING HEURISTICS OVER 2,000 OBJECT PACKING SEQUENCES

Method	Time (s)	Volume (cm^3)	Compactness	# Packed
Random[LB]	0.05	10,905	39.9%	27.54
DBL(BBox Order)	0.82	15,073	50.2%	37.17
HM (BBox Order)	0.83	15,496	51.7%	38.60
DBL(Balance)	1.90	15,117	50.3%	36.21
HM (Balance)	1.64	14,094	49.2%	38.43
Ours	**0.43**	**16,525**	**55.9%**	**39.88**
SDF-Minimization[UB]	1.86	16,897	56.8%	40.77

UB and LB denote upper and lower bounds in terms of compactness. Our method is fast and able to produce compact packing. See also fig. 1 for a plot on compactness vs. time.

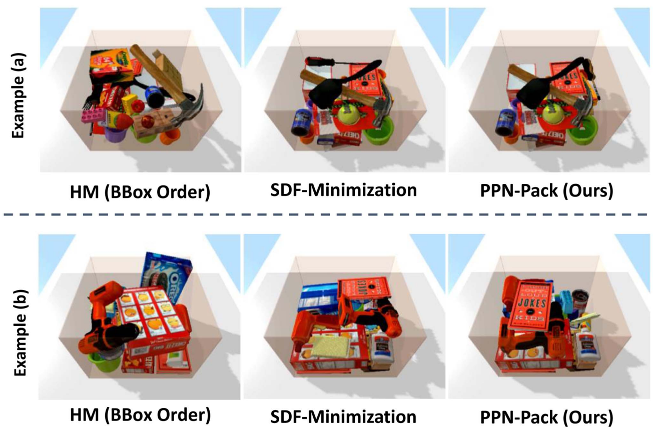


Fig. 5. Example results (a), (b) of packing 25 objects produced by different methods. We compare our PPN-Pack with HM (BBox Order) [9] and SDF-Minimization [11] (upper bound). Among all the methods, PPN-Pack runs the fastest and it can deliver comparable packing compactness as the upper bound; see also Table I.

0.2 cm in the z dimension and four types of object orientations on the xy-plane are considered, as in [11]. We set the gravity as -9.8 m/s^2. After packing each object into the container, we wait 0.25 seconds for the objects to stabilize before scanning the container's heightmap in the next packing step.

Evaluation metrics: Following [11], we evaluate the packing performance of different methods by (i) computation time (Time), i.e., the average time measured in seconds to choose an object and determine its placement; (ii) volume of packed objects (Volume), i.e., the total volume of objects packed inside the container, measured in cm^3; (iii) compactness, i.e., the total volume of packed objects divided by the volume of the minimum bounding box that encloses all the packed objects, and (iv) the number of packed objects (# Packed).

B. Comparison With Existing Heuristics

We first evaluate the sequential-packing performance on the 2,000 packing sequences to compare our method against the existing heuristics: Deepest Bottom-Left (DBL) [10] and heightmap-minimization (HM) [9]. Since the two packing heuristics are not designed for choosing the optimal object, we assist them in determining which object to be chosen by the bounding-box volume decreasing order (BBox Order) as in [9] and the size-balancing term (Balance) as in [11]. Besides, we compare with the SDF-Minimization heuristic [11], which exhaustively evaluates all placements, thereby providing an upper bound for our method. On the other hand, as a lower bound, we show the performance of a simple random placement (Random), which randomly selects an object in the buffer and randomly finds a feasible placement. Since this work aims for time efficiency, we focus on comparisons with fast heuristic-based methods.

From Table I, we can see that our method can greatly reduce the computation time while achieving compact packing, outperforming DBL and HM consistently for both the BBox Order and Balance settings. Specifically, it takes only 0.43 seconds to compute the object placement, consuming 47% to 77% less computation time compared to different versions of DBL and HM. Also, it can pack 6.6% more volume and 1.28 to 3.67 more objects with 4% higher compactness.

Compared with SDF-Minimization [11] (upper bound), which searches over all placements, our PPN-Pack achieves comparable packing compactness while greatly reducing 76.9% of computation time. Compared with Random (lower bound), we gain 55% in terms of packed volume, 16% in packing compactness, and 12.3 in number of objects.

Fig. 5 shows the packing results of our PPN-Pack in comparison with Heightmap-Minimization [9] (with BBox Order) and SDF-Minimization [11] on two packing sequences. For ease of visualization, we show the results of packing a set of 25 objects inside the same container. From the figure, we can see that our method achieves similar packing compactness as SDF-Minimization [11] but our method consumes far less computation time. Also, our results are more compact than those of Heightmap-Minimization while having a higher computational efficiency. More visual comparison results can be found in the supplementary material.

Although the proposed method requires a training procedure, it is a one-off process. Once the PPN is trained, it can help greatly reduce the computation time for numerous packing requests. In the absence of a trained network and the existence of few requests, heuristic methods remain useful.

C. Comparison With Reinforcement Learning

We also compare our method with a packing method based on hierarchical reinforcement learning [22] (HRL). To allow a quantitative comparison, we follow their setting, i.e., same container size and same dataset (kindly provided by the authors), with an easy subset and a hard subset divided according to the shape regularities, to sample 2,000 packing sequences with a length of 50 objects for evaluation. To train the PPN, we follow [22] to train two separate networks: one for the easy subset and the other for the hard subset. For the evaluation metrics, since a pre-trained model from HRL is not publicly available yet, we compare PPN-Pack with HRL by directly using the performance numbers in [22]. Note that we compare computation time, compactness, and number of packed objects, but not on packed volume, since packed volume is not reported in [22]. Also, our method is tested on an NVIDIA GeForce RTX 2080, while HRL is accelerated with an NVIDIA GeForce RTX

TABLE II
COMPARING WITH HRL [22] AVERAGED OVER 2,000 PACKING SEQUENCES

		Time (s)	Compactness	# Packed
Easy	**HRL**	0.97	45.6%	40.38
	Ours	**0.87**	**47.9%**	**44.16**
Hard	**HRL**	0.97	42.3%	37.28
	Ours	**0.93**	**42.4%**	**45.61**

TABLE III
ABLATION STUDIES

	Top-1%	Top-5%	Top-8%	Top-10%	Top-20%
w/o Rank & CCorr	57.9%	66.9%	70.2%	72.5%	80.8%
w/o Rank	62.1%	72.9%	77.0%	79.2%	86.1%
Full Model	**81.0%**	**94.9%**	**97.2%**	**98.0%**	**99.4%**

Removing ranking learning and/or the cross-correlation module lead(s) to performance drop when compared with our full model.

3090 GPU, whose TFLOPS is over 3.3 times of ours. Still, our method achieves a higher time efficiency than HRL.

Table II reports the experimental results, showing that our method uses less computation time and achieves a better packing performance on the same dataset: 10% less computation time on the easy subset and 4% less on the hard subset. Also, our method can pack 3.7 more objects on the easy subset and 8.3 more objects on the hard subset, while achieving higher packing compactness on both subsets. Besides, our training procedure is more efficient, taking only 3 GPU hours, while training HRL requires 16 GPU hours.

D. Ablation Studies

Next, we perform an ablation study on the major components in PPN-Pack, i.e., the ranking-learning formulation and the multi-resolution cross-correlation module. In detail, we iteratively remove these two components by the following strategies: (i) remove the ranking learning by formulating the problem as regression learning and training with the MSE loss; (ii) remove the cross-correlation module by concatenating the object with the container's feature maps instead.

As Table III shows, we evaluate the full model and the two ablated models for different hyperparameters K. From the results, we can see that upon removing the ranking learning, we observe a large drop of around 10% - 20% in the top-K% accuracy ("Full Model" v.s. "w/o Rank"). Then, further removing the cross-correlation module would result in a model performance drop of 5% - 7% ("w/o Rank" v.s. "w/o Rank & CCorr"). These performance drops show the contributions of the components to PPN-Pack.

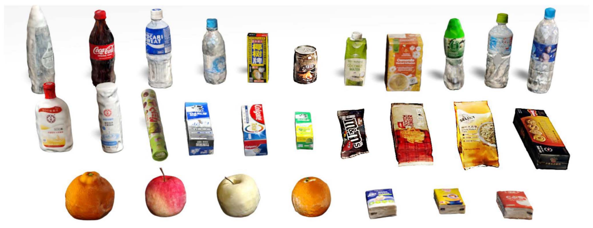
E. Handling Unseen Object Types

In this subsection, we study PPN-Pack's generalizability by evaluating its performance on 28 real-world objects unseen in the training phase, which are obtained by the Scan3D App. We evaluate the packing performance of PPN-Pack ("PPN-Pack (Ours)") and compare it with the upper bound of searching over all placements ("Search all") and lower bound of random search ("Random"). We set K=5, following the setting of "Sequential packing"; see Section IV-A.

TABLE IV
GENERALIZABILITY OF OUR PPN-PACK TOWARDS UNSEEN OBJECTS

	Time (s)	Volume (cm^3)	Compactness	# Packed
Random	0.37	19,029	39.5%	27.54
PPN-Pack (Ours)	0.45	22,993	47.3%	33.69
Search all	1.58	25,001	49.8%	34.26

The packing performance is average over 400 packing cases. We visualize the 28 unseen objects in the figure below.



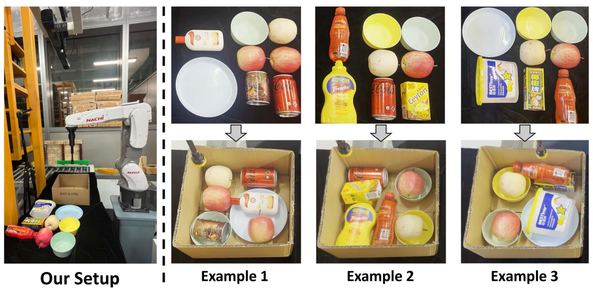


Fig. 6. Three real-world packing examples produced by our PPN-Pack using the NACHI MZ07 Robot. Our method successfully packs all the real-world objects into the container and greatly reduces the average packing computation time for each object from 1.6 seconds to 0.6 seconds.

Table IV summarizes the experimental results and shows also the 28 unseen objects. PPN-Pack generalizes well to the unseen objects by clearly surpassing "Random" with a comparable performance against "Search all". Compared with "Random", it can pack 20% more object volume, greatly improving the packing compactness by over 7.8%, and pack 6.1 more objects. Compared with "Search all", PPN-Pack can greatly reduce the computation time by 71%, while maintaining similar packed volume, packing compactness, and number of packed objects.

F. Extension to Physical Platform

Lastly, we demonstrate the applicability of PPN-Pack to real-world robotic packing scenes using the NACHI MZ07 Robot with a suction cup. Fig. 6 shows three example cases of packing eight general objects into a container of size 30 cm $\times$ 30 cm $\times$ 16 cm. Real-scanned point clouds, with most objects unseen in the training dataset (except the mustard bottle in Example 2), are used. A Photoneo camera captures the top-down point clouds of the container and the candidate objects. Iterative point matching estimates the pose of each candidate object to be packed, yielding its oriented mesh template for picking and packing planning. Heightmap pairs for objects are derived from the oriented mesh, while the container heightmap is obtained from the real-scanned point clouds. We set a buffer size of 2 and set K as 20. Our PPN-Pack reduces the average computation time per object from 1.6 to 0.6 seconds, representing a 62.5%

reduction in computational effort. Even with real-scanned point clouds and nearly all objects unseen in training, our PPN-Pack produces compact packing results with fast computation time, demonstrating strong generalizability. Please refer to the supplementary video for more results.

V. Conclusion

We presented PPN-Pack, a new learning-based approach for efficient computation in robotic object packing. Our approach leverages a placement proposal network (PPN) trained to prioritize compact placements by learning their potential compactness. Particularly, a multi-resolution cross-correlation module is proposed to model the geometric correlation between the container and the object to be packed. Experiments on 2000 packing sequences, featuring diverse everyday objects, demonstrate that our method can greatly reduce the packing computation time by 76%. Also, it exhibits good generalizability on unseen objects and real-world scenarios.

Although effective, the proposed method has limitations. First, it requires a network training procedure of approximately 3 GPU hours, though this is a one-off process. Secondly, it primarily differentiates optimal from non-optimal placements. In the future, we'll further rank the near-optimal placements to refine the proposal ranking and enhance the placement search efficiency. Also, while the approach uses a single packing objective, future research could explore integrating multiple objectives and addressing ranking uncertainties.

Acknowledgment

The authors would like to thank Yi Dong for his valuable contributions in scanning and processing real-world objects for use in several experiments.

References

[1] H. Zhao, Z. Pan, Y. Yu, and K. Xu, "Learning physically realizable skills for online packing of general 3D shapes," *ACM Trans. Graph.*, vol. 42, no. 5, pp. 1–21, 2023.

[2] R. Hu, J. Xu, B. Chen, M. Gong, H. Zhang, and H. Huang, "TAP-Net: Transport-and-pack using reinforcement learning," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–15, 2020.

[3] Q. Cui, V. Rong, D. Chen, and W. Matusik, "Dense, interlocking-free and scalable spectral packing of generic 3D objects," *ACM Trans. Graph*, vol. 42, no. 4, pp. 1–14, 2023.

[4] Z. Yang, Z. Pan, M. Li, K. Wu, and X. Gao, "Learning based 2D irregular shape packing," *ACM Trans. Graph.*, vol. 42, no. 6, pp. 1–16, 2023.

[5] Q. Zhu et al., "Learning to pack: A data-driven tree search algorithm for large-scale 3D bin packing problem," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 4393–4402.

[6] S. Yang et al., "Heuristics integrated deep reinforcement learning for online 3D bin packing," *IEEE Trans. Automat. Sci. Eng.*, vol. 21, no. 1, pp. 939–950, Jan. 2024.

[7] A. V. Puche and S. Lee, "Online 3D bin packing reinforcement learning solution with buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 8902–8909.

[8] J. Jia, H. Shang, and X. Chen, "Robot online 3D bin packing strategy based on deep reinforcement learning and 3D vision," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2022, pp. 1–6.

[9] F. Wang and K. Hauser, "Stable bin packing of non-convex 3D objects with a robot manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8698–8704.

[10] K. Karabulut and M. M. İnceoğlu, "A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method," in *Proc. Int. Conf. Adv. Inf. Syst.*, 2004, pp. 441–450.

[11] J.-H. Pan et al., "SDF-Pack: Towards compact bin packing with signed-distance-field minimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 10612–10619.

[12] A. G. Ramos, J. F. Oliveira, J. F. Gonçalves, and M. P. Lopes, "A container loading algorithm with static mechanical equilibrium stability constraints," *Transp. Res. Part B, Methodological*, vol. 91, pp. 565–581, 2016.

[13] K. Kang, I. Moon, and H. Wang, "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem," *Appl. Math. Comput.*, vol. 219, no. 3, pp. 1287–1299, 2012.

[14] T. G. Crainic, G. Perboli, and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *Informs J. Comput.*, vol. 20, no. 3, pp. 368–384, 2008.

[15] X. Liu, J.-M. Liu, A.-X. Cao, and Z.-L. Yao, "HAPE3D–a new constructive algorithm for the 3D irregular packing problem," *Front. Inf. Technol. Electron. Eng.*, vol. 16, no. 5, pp. 380–390, 2015.

[16] C. Lamas-Fernandez, J. A. Bennell, and A. Martinez-Sykora, "Voxel-based solution approaches to the three-dimensional irregular packing problem," *Operations Res.*, vol. 71, no. 4, pp. 1298–1317, 2023.

[17] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1021–1043, 2012.

[18] Z. Yang et al., "PackerBot: Variable-sized product packing with heuristic deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5002–5008.

[19] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, "Online 3D bin packing with constrained deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 1, pp. 741–749.

[20] H. Zhao, Y. Yu, and K. Xu, "Learning efficient online 3D bin packing on packing configuration trees," in *Proc. Int. Conf. Learn. Representations*, 2021.

[21] J. Zhang, B. Zi, and X. Ge, "Attend2Pack: Bin packing through deep reinforcement learning with attention," in *Proc. Int. Conf. Mach. Learn. Workshops*, 2021.

[22] S. Huang, Z. Wang, J. Zhou, and J. Lu, "Planning irregular object packing via hierarchical reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 8, no. 1, pp. 81–88, Jan. 2023.

[23] L. Hui, L. Wang, L. Tang, K. Lan, J. Xie, and J. Yang, "3D siamese transformer network for single object tracking on point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 293–310.

[24] A. Zeng et al., "Transporter networks: Rearranging the visual world for robotic manipulation," in *Proc. Conf. Robot Learn.*, 2020, pp. 726–747.

[25] H. Huang, D. Wang, R. Walters, and R. Platt, "Equivariant transporter network," in *Proc. Robotics, Sci. Syst.*, 2022.

[26] H. Doughty, W. Mayol-Cuevas, and D. Damen, "The pros and cons: Rank-aware temporal attention for skill determination in long videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7862–7871.

[27] J.-H. Pan, J. Gao, and W.-S. Zheng, "Adaptive action assessment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8779–8795, Dec. 2022.

[28] W. Li, J. Lu, A. Wuerkaixi, J. Feng, and J. Zhou, "MetaAge: Meta-learning personalized age estimators," *IEEE Trans. Image Process.*, vol. 31, pp. 4761–4775, 2022.

[29] H. Liu, J. Lu, J. Feng, and J. Zhou, "Ordinal deep learning for facial age estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 2, pp. 486–501, Feb. 2019.

[30] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set," *IEEE Robot. Automat. Mag.*, vol. 22, no. 3, pp. 36–52, Sep. 2015.

[31] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robot. Automat. Lett.*, vol. 1, no. 2, pp. 1179–1185, Jul. 2016.

[32] J. Huang, H. Su, and L. Guibas, "Robust watertight manifold surface generation method for ShapeNet models," 2018, *arXiv:1802.01698*.

[33] K. Mamou, E. Lengyel, and A. Peters, "Volumetric hierarchical approximate convex decomposition," in *Game Engine Gems*, vol. 3, 2016, pp. 141–158.

[34] E. Coumans and Y. Bai, "PyBullet, a python module for physics simulation for games, robotics and machine learning," 2016–2021. [Online]. Available: http://pybullet.org