Delft University of Technology

MSc Thesis

Learning object-object affordances for placement-position prediction in an item packing task

---

Author:
Besim Bay
4672879

Supervisors:
Prof. Dr. Jens Kober
Dr. Ravi Prakash

A Thesis submitted in fulfillment of the requirements for the degree of MSc Robotics

in the

Department of Cognitive Robotics

September 30, 2022

Contents

Abstract **1**

1 Introduction **2**
1.1 Robotic item packing . . . 2
1.2 Related work . . . 3
1.3 Research objective . . . 4
1.4 Thesis outline . . . 5

2 Methods **6**
2.1 Data and methodology . . . 6
2.2 Data generation . . . 7
2.3 Simulated interaction trials . . . 9
2.4 Network architecture . . . 9
2.4.1 Input and output . . . 9
2.4.2 Architecture . . . 10
2.4.3 Loss and training . . . 11
2.5 Item packing pipeline . . . 11

3 Results **13**
3.1 Experimental setup . . . 13
3.2 Performance metric . . . 13
3.3 Single items . . . 16
3.4 Novel items . . . 19
3.5 Real data . . . 21
3.6 Packing a list of items . . . 23

4 Discussion and Conclusion **30**
4.1 Discussion . . . 30
4.2 Conclusion . . . 31

Bibliography **33**

Appendix A **35**

Abstract

Grocery e-commerce has been rapidly increasing in recent years, posing a new challenge for retailers as groceries, unlike other goods, have a limited shelf life. Thus, customers expect their orders to arrive quickly and undamaged. Currently, most processes between a customer placing an order and the delivery are performed manually in a warehouse making it labor-intensive and time consuming. The use of autonomous robots in these environments can help improve operational efficiency and productivity while at the same time reducing labor costs and accidents.

One specific process important for maintaining the desired quality of the customer's ordered items and fast delivery is item packing. For an autonomous robot to pack items independently, it must learn and predict possible placement positions that are both geometrically feasible and semantically plausible so that items maintain their desired quality. Retail-related environments are heterogeneous environments containing many different items. Thus, instead of the robot knowing what the item is, it is more beneficial to know how an item can interact with a scene. This entails that robots must learn interactions between items and the scene, which are called affordances, to place items in plausible positions. These so-called object-object affordances describe how an object can interact with another object.

In this thesis, object-object affordances are learned to predict where to place items inside a box in an item packing task. With the use of a simulator called SAPIEN, item packing is simulated, and large-scale interaction data is generated. A model is then trained to predict a placement position by giving as input a complete pointcloud of the item to be placed and a partial pointcloud of the scene. An item packing pipeline is then built that can pack items using the trained model. Several item packing experiments are performed to pack single items and a list of items. The results show that the model successfully learns the semantic relationships between objects resulting in packing items in stable and plausible positions. Several other experiments are performed to evaluate whether the model is generalizable to novel items and real pointcloud data. Results show that the model successfully predicts where novel items should be placed. The model can also adapt to real pointcloud data of a box and predict where items should be placed in a real box containing real items.

Keywords: Item packing, object-object affordances, pointcloud, SAPIEN simulator

1. Introduction

1.1 Robotic item packing

Over recent past years, there has been a rapid increase in grocery e-commerce, and customers will increasingly shop for groceries online in the years ahead [1]. However, groceries have, unlike other goods, limited shelf life. This means that when a customer orders groceries, they expect same-day delivery of fresh, high-quality produce. This poses a challenge for retailers as there is a necessity to deliver orders as fast as possible while maintaining the quality of easily damaged fresh items. In a retail warehouse, there are three main processes between a customer placing an order and the delivery: order picking, item packing, and delivery. In figure 1.1, these processes are shown.

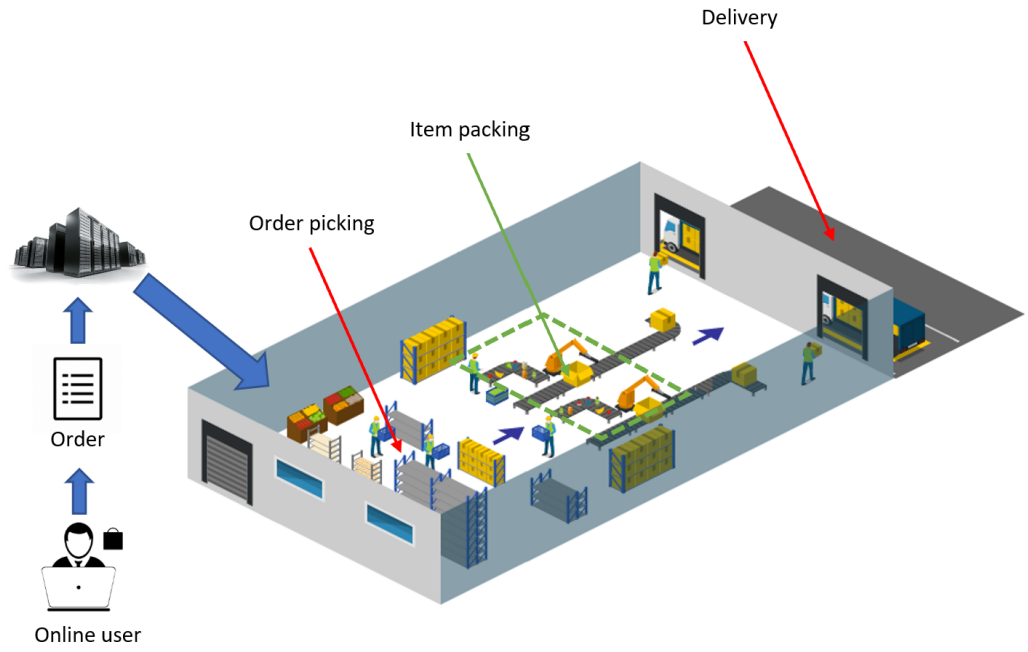


Figure 1.1: Processes between a customer placing an order and delivery. Envisioned robotic item packing is highlighted with a green dashed rectangle.

Currently, these three steps are often performed manually, making them labor-intensive and time consuming. With the help of autonomous robots, these labor-intensive tasks can be reduced, resulting in lower accidents and labour cost. At the same time, operational efficiency and productivity can be increased. While there has been much research into autonomous order picking using mobile manipulators and delivery using self driving vehicles [2], autonomous item packing is relatively unexplored. Item packing must be performed carefully to maintain the desired quality of products as

there are easily damaged items meaning that we can not place items randomly. First of all, items must be placed in stable positions so that they do not tip over and get damaged. Secondly, there are items that should preferably not be placed around other items. For example, a soft item like bananas should not be placed next to a hard item like bleach cleansers as it can damage the bananas. Another example is placing a hard item like a sugar box on top of strawberries which is also not preferred. Not only must the items be placed in stable positions that have enough space, i.e., geometrically feasible positions. Items must also be placed around other items having the same categorical labels, i.e., semantically plausible positions. Thus, for an autonomous robot to pack items independently, it must learn and predict possible goal positions that afford placement by considering both spatial and semantic relationships. One possible approach is incorporating knowledge of all items, including a fixed position in the box. However, retail-related environments are heterogeneous environments containing many different items, and the combinations of items to be packed are never the same. This approach thus makes it very hard to scale, and incorporating such knowledge does not capture the essence of an item packing task. Instead of the robot knowing what the item is, it is more beneficial to know how an item can interact with a scene. Robots must learn interactions between items and the scene, which are called affordances, to place items in their correct position. Using these object-object affordances, certain priors for interactions between items can be learned based on the shape and appearance of the items.

1.2 Related work

Current research on robotic item packing often focuses on the grasping part during item packing. In these works, the focus lies on detecting objects and computing grasp poses, after which the items are picked up and dropped randomly in a bin[3]. These works do not focus on where to place the item after picking them up. Most of the research that do focus on where to place items treat the problem as a bin packing problem in which the objective is to pack multiple of the same item in a box tightly together[4]. They do not consider various types of items and whether certain items can be placed nearby each other or not, i.e., the semantic relationships between items. The works that use affordance-based solutions to learning problems that can be useful in item packing also focus primarily on grasping. Before discussing how affordances are being used, two types of interaction affordances and their formal definitions are described below:

Agent-object interaction affordances describe how an agent might interact with an object. This can be subdivided into three popular categories: Robot-object, Human-object, and Hand-object. An agent-object affordance model accounts for a high-level behavior modeling approach that learns the relationships between the agent, its actions, and their effect on objects [5]. For a given object, an affordance A_F represents a subset of the state-action space ($A_F \subset S \times A$), that leads to the intended effect. Once discovered, affordances might offer a kind of generalization across different objects of the same class. High-level decision-making can utilize then affordances to achieve desired goals/transitions in the environment in an efficient and effective manner.

Object-object affordances describe how an object can interact with another object. An example scenario is fitting an item inside a drawer. We humans can instantly understand if the object fits after a glance at the scene. This is because we humans learn certain priors for interactions based on the shape and functionality of the objects. However, robots do not have this capability, and object-object interaction affordances provide understanding in these scenarios.

There is a lot of research on agent-object interaction affordances in prior literature. Primarily in grasping affordances in both robot-object and hand-object interactions. [6, 7, 8] try to anticipate the success of a given grasp by formulating grasping as a visual affordance detection problem. An affordance detection model is developed to predict objects' graspable areas from images or point cloud data for a robot gripper. [9, 10] extend grasp affordances from a robotic gripper to human hands object interactions. [11] predicts how a human would grasp one or several objects, given a single RGB image of these objects. The disadvantage of most of this research is that they heavily depend on annotated data for training. Therefore, recently some research has been developed on obtaining visual affordances from human demonstration videos in a weakly-supervised manner. [12] developed a model to extract features embedded in demonstration videos of humans interacting with an object which allows predicting the interaction region and the action label on an image of the same object. [13] proposes an approach to learning human-object interaction hotspots from videos of a human interacting with an object. Their approach allows inferring a spatial hotspot affordance map indicating how an object would be manipulated in a potential interaction. While these works focus on grasping items using affordances, the use of object-object affordances to learn where to place items remains a challenge.

There is preliminary research on stable placements of items using object-object affordances. In [14], the authors develop an affordance model that can predict positions in a scene that afford stable placements to take place on different objects. Given a point cloud of the object and a partial scene scan, their model can predict positions on the scene that afford a placement to take place on the object. In [15], the authors develop an affordance model that can predict stable placement positions for various objects in a scene. Their model can predict stable placement positions using point cloud data of the object and a placement area. That model learns by sampling placements, for which several features are computed that indicate stability and support. A supervised learning technique is then used to map these features to good placements. These two works focus on predicting stable positions, which is necessary for item packing as well. However, for packing items, these works are insufficient as items can not be placed in only stable positions, as mentioned earlier.

1.3 Research objective

This thesis considers the problem of learning object-object affordances to predict where to place grocery items in a box during an item packing task. Given as input a complete 6D pointcloud of the acting item to be placed along with a partial 6D pointcloud of an existing scene, with each pointcloud consisting of XYZ coordinates and corresponding RGB colors, the objective is to produce a point-wise affordance heatmap on the scene

that measures the likelihood for each point on the scene point cloud of successfully accomplishing the task and being a plausible placement. The item will then be placed on the predicted placement-position by picking the position of the point with the highest predicted value. Hereby, it is important to show that the predicted positions are geometrically feasible and semantically plausible. This means that the predicted positions must both provide stability and enough space. At the same time, items must be predicted to be placed in semantically plausible positions meaning that objects must be placed nearby other objects that possess similar properties. Using the SAPIEN physical simulator and 3D shape datasets called ShapeNet and YCB, item packing scenarios are simulated to generate data. By generating large-scale realistic interaction data, the model should learn the semantic relationships between objects and place the items accordingly. One important advantage of the proposed method is that it can reduce the search space for downstream planning tasks. Another advantage is that it can deal with arbitrary scenes meaning that the model can predict where an item should be placed in a box containing an arbitrary number of items. At the same time, the proposed method increases generalizability as it can deal with previously unseen items and real data. Lastly, the model takes a partial pointcloud of the scene as input, which depth cameras can relatively quickly obtain, like Intel RealSense, making the model suitable for deployment on a real robot.

The main contributions are:

- Create a simulation environment in SAPIEN and use objects from the YCB dataset to simulate item packing and collect synthetic interaction data.
- Create a pipeline to pack item(s) using the trained model.
- Show that the model learned spatial and semantic relationships between items resulting in successful packing of single items and a list of items.
- Show that the model is successful in adapting to novel items and real data.

1.4 Thesis outline

The remainder of the thesis is structured as follows: Chapter 2 will contain the methods section describing the used data, the method for data generation, the network architecture and the algorithmic item packing pipeline. Chapter 3 will include the results in which various experiments will be performed to evaluate the model's performance in packing items. It will show comparisons with the baseline model and results for novel items and real world data. Finally, chapter 4 contains the discussion and conclusions summarising the work done and suggesting potential extensions.

2. Methods

2.1 Data and methodology

The task of robotic item packing consists of packing retail-related various items inside a carton box. The model of the carton box is obtained from the ShapeNet dataset[16], a large-scale dataset of 3D shapes. The dimensions of the box are $0.4 \times 0.4 \times 0.3$ (m) and the carton box is always open. The items that are used come from the YCB dataset[17]. YCB is an object and model set designed for facilitating benchmarking in robotic manipulation. It contains various objects of daily life with different shapes and textures. The fourteen used items in this thesis come from the food-and kitchen item set. From the YCB dataset, the object files in a .OBJ format are obtained for the fourteen items that will be used.

To achieve the objective of packing items, first, large scale interaction data is generated. Then, a model is trained on the generated data after which an algorithmic pipeline is built that uses the trained model to pack item(s). In figure 2.1 a blockdiagram is shown containing three developed modules to achieve the goal of packing items: data generation, model training and packing item(s). In the subsequent sections, the three modules will be explained in detail.

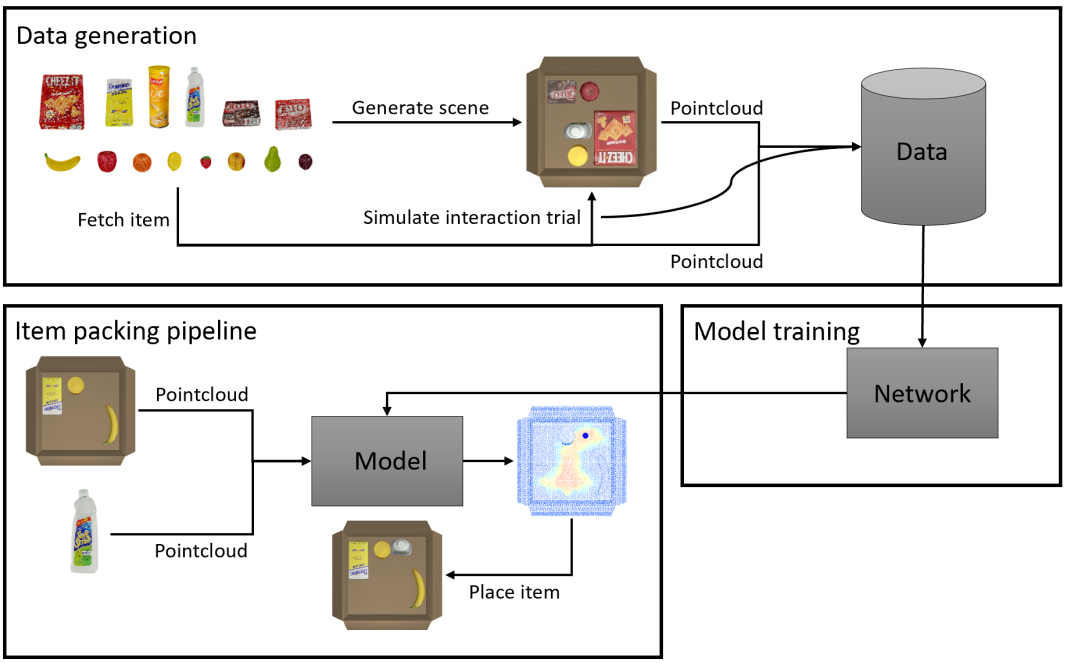


Figure 2.1: Block diagram containing the three developed modules: data generation, model training and item packing pipeline.

2.2 Data generation

To collect interaction data to train the model on, a physics based simulation called SAPIEN is used [18]. A simulator is used because synthetic data generation is faster, more flexible, and more scalable. Generating large-scale interaction data of placing items in boxes in real life would take a lot of time and effort, making the simulator more efficient. The simulator is equipped with a carton box from the ShapeNet dataset. In figure 2.2, the Sapien simulator window and the box are visualized. A camera is positioned above the box in the simulator through which the complete inside of the box can be observed. Fourteen grocery store items that are used are obtained from the YCB dataset, and the complete pointcloud models of these items are visualized in figure 2.3.

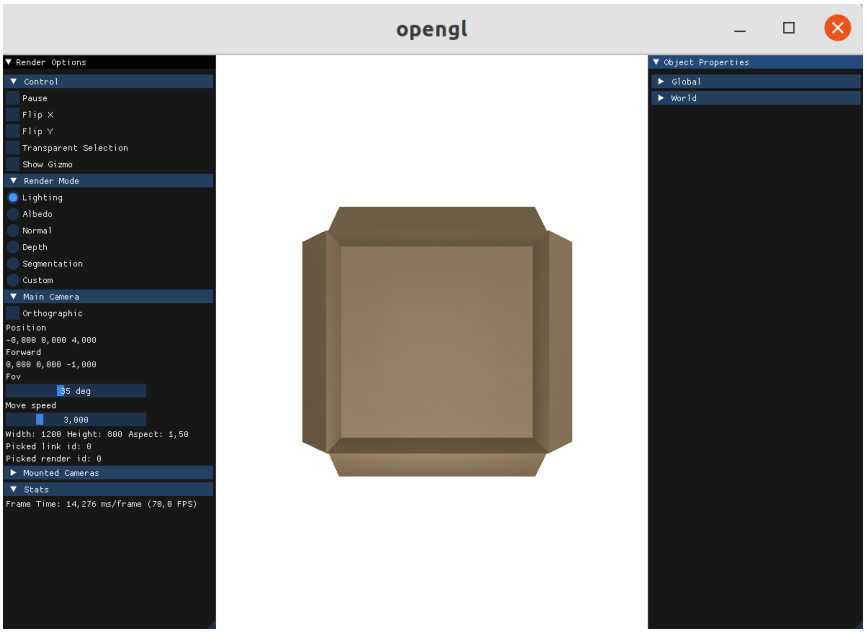


Figure 2.2: SAPIEN simulator setup

Figure 2.3: Pointclouds of the used items from YCB dataset

The steps that are taken to generate interaction data are described below. First, a list of scene objects to populate the box is generated by randomly sampling one to five objects selected from the YCB dataset. All of the available objects are labeled with four labels: hard/soft and stackable/nonstackable. An object is labeled as stackable if it can be stacked on top of another object stably and if it allows another item to be stacked

Objects	hard/soft	stackable/nonstackable
Cracker box	hard	stackable
Sugar box	hard	stackable
Chips can	hard	nonstackable
Bleach cleanser	hard	nonstackable
Pudding box	soft	stackable
Gelatin box	soft	stackable
Banana	soft	nonstackable
Apple	soft	nonstackable
Orange	soft	nonstackable
Lemon	soft	nonstackable
Strawberry	soft	nonstackable
Peach	soft	nonstackable
Pear	soft	nonstackable
Plum	soft	nonstackable

Table 2.1: Fourteen used item and their given labels for data generation.

on top of it. For example, items such as cracker boxes and sugar boxes are stackable. Items are labeled soft when they are sensitive and hard if they are solid. Items such as a bleach cleanser and a sugar box are labeled hard, while fruits such as bananas are labeled soft, for example. In table 2.1 an overview of all the used items and their corresponding labels is given.

The items in the scene object list are then sorted based on whether they are stackable. Stackable items are placed first. Every first item is placed in one of the four corners inside the box, just like most humans would place the first item. Every second item is automatically checked whether it belongs to the same hard or soft class; if it does, it is placed near the already placed item. If it does not belong to the same class, it is placed as far away as possible. This is accomplished by selecting the position on the bottom surface that maximizes the euclidean distance between the item to be placed and the previous item.

Every next item is placed around the already placed items belonging to the same class. If the item is stackable and the same item is already placed in the box, it is placed on top of it. In figure 2.4 several example generated scenes are shown. Once the scene is populated with scene objects, the camera in the simulator takes a single snapshot from above the scene to obtain a partial 6D scene pointcloud $S \in \mathbb{R}^{n\times 6}$ containing XYZ positions and RGB color information. Next, an acting item to be placed is fetched, and a complete poincloud $O \in \mathbb{R}^{m\times 6}$ is obtained. To capture the complete 6D pointcloud of the item, a function is created that uses functions from the Trimesh library to load the object file and sample m amount of points and their corresponding color values on the surface. The pointcloud of the item is transformed to the same camera base coordinate frame as the scene point cloud. m and n are defined to be 10000 meaning that the scene and item pointclouds are represented with 10000 points containing XYZ positions and RGB color values.

(a) Box with 1 item (b) Box with 3 items (c) Box with 5 items

Figure 2.4: Example scenes

2.3 Simulated interaction trials

Once a scene is generated, a placement interaction is simulated. For each interaction trial, a short hard-coded trajectory is executed for the fetched item to be placed. During this trajectory, the item to be placed is dropped from a certain distance above the box. Hereby, the trajectory is always in a straight line. The position at which the simulated interaction happens must be semantically plausible and therefore it is based on the categorical labels of the objects. If the item is a stackable object, the item is placed on top of another stackable item belonging to the same class. If it is a non-stackable item, it is placed nearby other items belonging to the same class. Furthermore, if it is a soft item, the item can be placed nearby other soft items but also on top of hard stackable items. These placements are achieved by keeping track of the positions of the previously placed scene objects and placing the item a certain distance away, depending on the object's size. Thus, semantically plausible placements are made by placing objects near other objects of the same categorical labels and placing stackable items on top of each other. The result of the interaction is either successful or failed for each interaction trial, measured by a metric. The metric measures whether the item is placed stably without tipping over by measuring the orientation before placement and after placement. If the orientation difference is within a certain bound, then it is geometrically feasible, and the outcome is successful. Thus, by placing items according to their labels and checking whether they remain stable after placement, interaction data is generated that is semantically plausible and geometric feasible. The position at which the item is dropped and whether the outcome is success or failure is saved and used during model training, which will be explained in the next section.

2.4 Network architecture

2.4.1 Input and output

The O2O-afford model developed in [14] is used to tackle this learning problem and taken as a baseline model. One disadvantage of this baseline model is that the focus lies only on predicting stable placement positions as described in section 1.2. This baseline model is trained on interaction data in which items are randomly placed on a surface and checked whether it remains stable or not. Thus, it is expected that the baseline model will only predict stable placement positions located on the bottom surface of the box

without taking into consideration semantic relationships. This disadvantage has been resolved by training the model on interaction data generated by the method described in section 2.2. Another disadvantage of the baseline is that it takes as input only a 3D partial pointcloud scan of the scene and a complete 3D pointcloud of the acting object to be placed containing only XYZ coordinates. This positional information only captures the shape of the object and the scene. Therefore, the model can only reason and learn about the shapes and geometry, which is useful for learning geometrically feasible placements. However, retail items consist of different colors, such as fruits having a single brighter color, while boxes and other items consist of multiple colors. Because items are placed in semantically plausible positions according to their categorical label, their color can be a good feature to help reason about the placement during training, as items with the same categorical labels have similar color properties. Therefore, this baseline model has been extended further by incorporating additional RGB color to the inputs of the scene's and acting object's pointcloud. Thus, rather than the points in the pointcloud only containing XYZ coordinates, in the developed model, the points of the input pointclouds consist of XYZ coordinates and RGB color values, making the dimension 6D. The model has been adjusted to cope with this new input dimension.

The output of the model is a per-point affordance heatmap $A \in [0,1]^m$ over the scene point cloud S. Thus, for each point $p_i \in S$, the model will predict a likelihood $a_i \in [0,1]$ of the acting object O being a good placement at p_i which is semantically plausible and geometrically feasible.

2.4.2 Architecture

The model works as follows: The input to the model is a 6D pointcloud of the scene ($S \in \mathbb{R}^{n\times 6}$) and item ($O \in \mathbb{R}^{m\times 6}$). Two separate PointNet++ [19] networks are trained to extract per-point features for the scene and item. PointNet++ is a network that builds further upon PointNet [20]. Pointnet is a network that learns each point's spatial encodings and then aggregates all individual point features to a global point cloud feature map. It fails to capture local structures and generalize to complex models. PointNet++ applies the PointNet network recursively on partitionings of the input pointcloud. It uses additional learning layers to combine features from multiple scales. It extracts local features from partitioning, which are further grouped into larger units and processed to produce higher-level features. This happens recursively to obtain the feature of the whole pointcloud. In figure 2.5 the architecture of the network can be seen. For the scene, the network extracts a per-point feature map $F_S \in \mathbb{R}^{n\times f_1}$. For the object, the network extracts a per-point feature map $F_O \in \mathbb{R}^{m\times f_2}$ and a global feature $F_g \in \mathbb{R}^{f_g}$.

After the per-point feature maps are extracted, the model samples k number of points $(p_1, p_2, ..., p_k)$ on the scene pointcloud using furthest point sampling(FPS). FPS is a greedy algorithm that iteratively samples points from a pointcloud. It starts sampling from a random point. Each iteration samples from the rest points that are the farthest from the set of sampled points. Once the points have been sampled, the acting object is used a kernel to slide over the points to perform a point-wise convolution operation. In detail, the acting object is stridden over the sampled scene points to query the scene feature map over the points of the acting object pointcloud, resulting in a scene feature map $F_{S|O,p_i} \in \mathbb{R}^{m\times f_1}$. The scene feature map and extracted object feature map is

then concatenated resulting in an aggregated feature map at every sampled point p_i $F_{SO|O,p_i} \in \mathbb{R}^{m\times(f_1+f_2)}$. A PointNet network is then used to perform a convolution to obtain the sampled point features $F_{SO|p_i} \in \mathbb{R}^{f_3}$. For every sampled point, the sampled point feature $F_{SO|p_i} \in \mathbb{R}^{f_3}$, the global object feature $F_g \in \mathbb{R}^{f_g}$ and the scene point feature $F_{S|p_i} \in \mathbb{R}^{f_1}$ is aggregated and fed into an MLP with a sigmoid activation function to obtain an affordance labeling $a_i \in [0, 1]$ for every sampled point p_i. The per-point affordance labeling for the sampled points is then interpolated back to all points on the scene pointcloud to obtain a point-wise affordance heatmap over the scene point cloud.

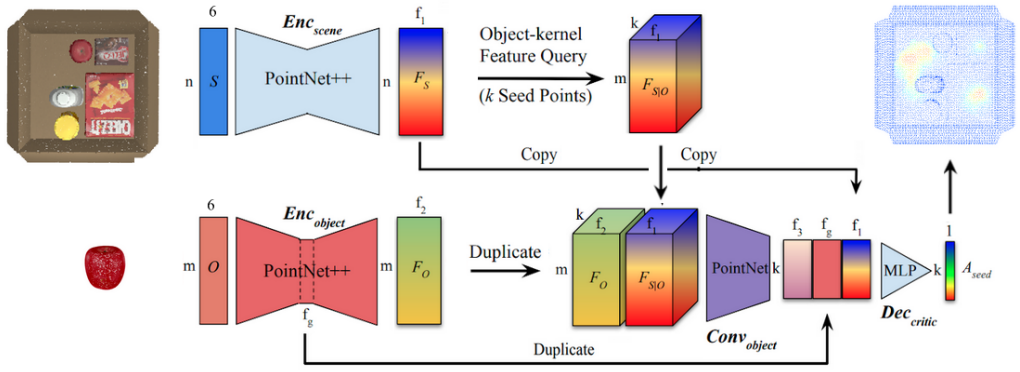


Figure 2.5: Network architecture, adapted from fig. 3 [14]

2.4.3 Loss and training

The model is supervised by the simulated interaction trials, i.e., the position with a successful or failed outcome. The standard binary cross-entropy loss between the outcome of the position during the interaction trial and the predicted value for this position is used to update the weights. During data generation, the scene and acting objects are selected randomly from available objects. The acting objects are equally sampled during data generation. It is empirically found that having around 5000 data samples is essential for successful training. As mentioned earlier, $n = m = 10000$ is used and $f_1 = f_2 = f_3 = f_g = 64$. Ten epochs are used during training and it is performed on a GeForce RTX 3050 Mobile GPU graphics card.

2.5 Item packing pipeline

The trained model is used to make predictions and pack items on the position with the highest predicted value. This is performed by creating a pipeline through which the trained model can be used to pack items. The pipeline will be explained using pseudocode, and in algorithm 1, the pseudocode for the pipeline is shown. The process of the pipeline is explained below with each of the numbers below corresponding to the line number in pseudocode 1.

1. , 2. First, the user can specify the scene object(s) to populate the box and the acting object(s) to be placed. If not specified, it will be sampled randomly.

3. Once the scene objects are specified, the scene will be generated by placing the scene objects one by one, as described in section 2.2.

6. Once the scene is generated, the script will loop over the acting objects to be placed in the specified order and perform the following steps:

7. For each acting object, first, the scene pointcloud is obtained using in-built functions from the SAPIEN simulator. An inbuilt function called get_observation() uses the camera in the simulator to take a snapshot of the scene to capture both the depth and color. These two are merged together to obtain a pointcloud in which the points contain both position and color.

8. Using a created function called generate_pointcloud() that takes the acting object name as an argument, the complete pointcloud for the acting object is obtained. The steps performed in this function are as follows: For every item used from the YCB dataset, the object file is provided in a .OBJ format with its colormap. Then, using the Trimesh library, the acting object is loaded into a Trimesh object. Finally, with the use of another function from the Trimesh library called sample_surface(), a preferred number of points are sampled on the surface of the mesh to obtain a complete pointcloud of the object.

9. Once the partial pointcloud of the scene and complete pointcloud of the object are obtained, they are fed as input to the model. The model outputs a prediction in which every point of the scene pointcloud is labeled with a value between 0 and 1, with a higher value indicating a better placement.

10. To find the position with the highest value, the index of the point with the highest predicted value is found, and the corresponding position of the point on the scene is found.

12. Finally, using an inbuilt function from the SAPIEN simulator called load_object(), the item is placed in the predicted position.

Algorithm 1 Pseudo code for placing a list of items

```
input: ActingObjects
input: SceneObjects
for SceneObject in SceneObjects do
    place in box
end for
for ActingObject in ActingObjects do
    scene_pc = cam.get_observation()
    acting_object_pc = generate_pointcloud(ActingObject)
    pred_result = network.inference_whole_pc(scene_pc, acting_object_pc)
    index_pred_pos = argmax(pred_result)
    pred_pos = scene_pc[index_pred_pos]
    load_object(ActingObject, pred_pos)
end for
```

3. Results

3.1 Experimental setup

Once the model has been trained, it is evaluated using the SAPIEN simulator again with the box and items from ShapeNet and the YCB dataset, respectively. Several item packing experiments have been performed using the item packing pipeline to evaluate the performance. The performed experiments are listed below:

(i) Single known item in a scene: In these experiments, the pipeline is used to predict and place a single known item in a box that already contains items. A comparison will made with the baseline model to evaluate the differences.

(ii) Single unknown item in a scene: In these experiments the generalizability to novel items is evaluated by using a novel item that the model has not seen during training and predicting where it needs to be placed in a box that contains items.

(iii) Single known item in a real scene: in these experiments, the generalizability to real data is evaluated by using pointclouds of a real box captured by a depth camera. A single item is used to predict where it needs to be placed in a real scene.

(iv) List of items in a scene: in these experiments the model's performance on packing a list of items is evaluated. Various number of arbitrary items are predicted and placed in an empty box. A comparison will be made with the baseline model as well to evaluate the differences.

The experiments and their results will be discussed in sections 3.3, 3.4, 3.5 and 3.6, respectively.

3.2 Performance metric

As described in section 1.3 containing the research objective, the model should make predictions that are both geometrically feasible and semantically plausible. Thus, the model should have learned semantic relationships during training and predict that items must be placed on stable positions that lie nearby items belonging to the same categorical label(class). For example, if the item to be placed is a chips can and if there are already chips cans and bleach cleansers in the box, the model should predict that it must be placed nearby them. To evaluate the performance of the model, two types of evaluations will be performed. Firstly, empirical testing by observing the predicted heatmaps together with the predicted position (position with the highest predicted value) and secondly, a quantitative analysis by using a performance metric to evaluate whether the predicted positions are semantically plausible.

A qualitative evaluation is performed by observing where the red colored regions are located on the predicted heatmaps. The model should label positions lying closer to items belonging to the same class with a higher value resulting in red colored regions lying close to items belonging to the same class.

A quantitative evaluation is performed by calculating whether the predicted positions on which the items will be placed lie closer to items it belongs to. This is performed by keeping track of the center positions of the placed scene objects in the box and calculating the euclidean distance between the predicted position(x_p, y_p) and each scene object. If the scene contains hard and soft scene objects, the minimum distance to the object of the same class (MS) and minimum distance to object of a different class (MD) is calculated. This is performed by equations 3.1 and 3.2 for MS and MD, respectively.

$$MS = \min \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2} \quad \text{for} \quad i = 1, 2, .., SC \tag{3.1}$$

$$\text{with SC = \# scene objects of same class}$$

$$MD = \min \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2} \quad \text{for} \quad i = 1, 2, .., DC \tag{3.2}$$

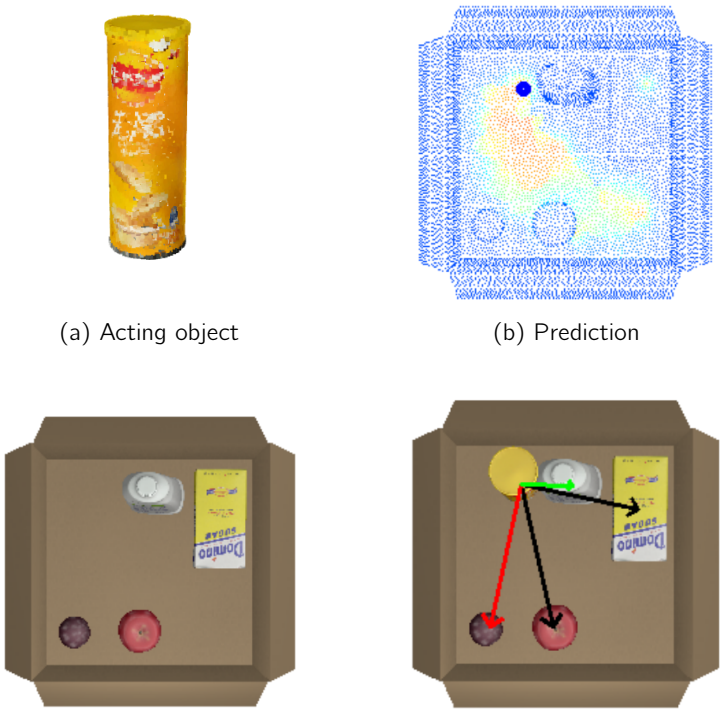
$$\text{with DC = \# scene objects of different class}$$

If the model makes semantically plausible predictions, the predicted positions for the items should lie closer to other items belonging to the same class compared to items belonging to a different class. This means that the minimum distance to an object of the same class should be smaller than the minimum distance of an object of a different class. Thus, the following equation must hold:

$$MS < MD \tag{3.3}$$

If this equation holds, it means that the model predicts positions that lie closer to items belonging to the same class compared to items belonging to a different class resulting in semantically plausible predictions. It is important to note that the class refers to the hard/soft label given to the items during data generation shown in table 2.1.

In figure 3.1, the result is shown for a chips can in a scene that already contains four items. In the box there are two hard items being the bleach cleanser and sugar box and two soft items. The model should predict that a hard item like a chips can must be placed nearby the other hard items. In figure 3.1b the predicted heatmap is shown with the position with the highest predicted value highlighted with a dark blue dot. In figure 3.1d the distances between the scene objects and the predicted position on which the chips can is placed are visualized with arrows. The MS is represented with the green colored arrow and the MD is represented with a red colored arrow the image. This color labeling of MS as green and MD as red will be valid for all other cases in the remaining sections. In table 3.1, the calculated values are shown. It can be observed that the MS is significantly smaller than MD. To verify that the model is indeed predicting correctly, the model is tested ten times on the same acting object and similar scenes and the values for the mean and standard deviations are computed.



(a) Acting object (b) Prediction

(c) Scene (d) Distances

Figure 3.1: Prediction result for a chips can. The model takes as input the pointcloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively.

	MS (in m)	*MD* (in m)
Figure 3.1	0.109	0.309
Mean	0.135	0.291
SD	0.036	0.020

Table 3.1: Calculated distances MS and MD for the experiment in figure 3.1. The experiment has been performed ten times and the mean and standard deviation for each distance is computed.

In appendix A, figures 1 and 2 several other prediction results for a chips can and similar scene are shown. The mean value for *MS* is significantly smaller than for *MD*, thus equation 3.1 is satisfied. The low standard deviations show that the model is repeatedly successful in predicting that a chips can should be placed closer to the other two hard items than to the soft items; there is low variance in the distances MS and MD over ten trials. It can thus be concluded that the model is successful in predicting stable semantically plausible positions for the chips can in this scene.

3.3 Single items

This thesis builds upon the O2O-afford model that focuses on predicting stable placement positions. The O2O-afford model is therefore taken as a baseline and comparisons will be made between the predictions made by the baseline and the developed model. To compare the performance of both models, the following experiments have been performed: Several scenes are generated, each containing 1 to 5 scene objects inside the box. For several acting objects, predictions are made with both the developed model and the baseline O2O-afford model and compared using the metric explained in section 3.2. It is expected that the predictions of the model in this thesis are, first of all, geometrically feasible and, secondly, semantically plausible. In contrast to the developed model, the baseline model only takes into consideration geometric feasibility. Their model is trained on interaction data in which items are randomly placed on the surface and checked whether it remains stable or not. Thus, it is expected that the baseline model will only predict stable placement positions located on the bottom surface of the box without taking into consideration semantic relationships that are imposed by the method used in section 2.2.

Figure 3.2 shows the results of the model and the baseline model for the same acting object; bleach cleanser in a scene already containing three items. The scene contains two hard items being the sugar box and banana and one soft item; the banana.

	MS (in m)	MD (in m)
Figure 3.2c	0.259	0.174
Mean	0.195	0.206
SD	0.075	0.029

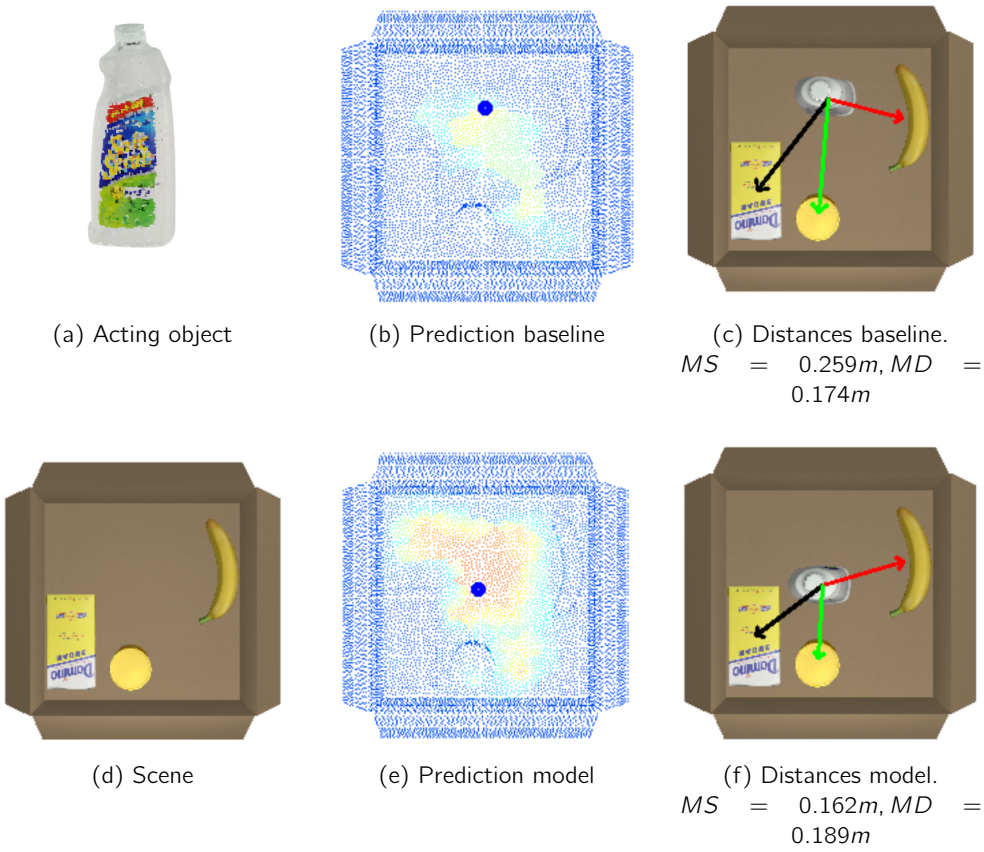
(a) Baseline

	MS (in m)	MD (in m)
Figure 3.2f	0.162	0.189
Mean	0.182	0.244
SD	0.055	0.069

(b) Model

Table 3.2: Calculated distances for the case in figure 3.2

Figure 3.2b and 3.2e show the prediction made by the baseline model and developed model with the highest predicted position marked with a dark blue dot. It can be observed that the heatmap of the baseline only consists of light yellow colored regions on the box, meaning that the baseline had difficulty in making predictions with high confidence. The baseline model had difficulty recognizing stable positions inside the box resulting in the predictions having a low value. If the baseline model had performed better, the empty spaces that provided enough space for the bleach cleanser would have been colored red. This is observable in the heatmap of the developed model. In contrast to the baseline, there are regions that are colored red inside the box positioned around the two other hard items sugar box and chips can box. Figure 3.2c and 3.2f show the placement of the bleach cleanser on the position with the highest predicted value for the baseline and developed model together with the distances to the scene objects represented as arrows. For the the baseline, the distance to the banana is smaller than the distance to the chips can which is undesired. The model is successful in learning that a bleach cleanser should be placed closer to the hard items than to the soft banana. The baseline and developed model are tested ten times on the same acting object and similar scenes to obtain the mean values and standard deviations. In appendix 4.2, figures 3 and 4



(a) Acting object

(b) Prediction baseline

(c) Distances baseline. $MS = 0.259m, MD = 0.174m$

(d) Scene

(e) Prediction model

(f) Distances model. $MS = 0.162m, MD = 0.189m$

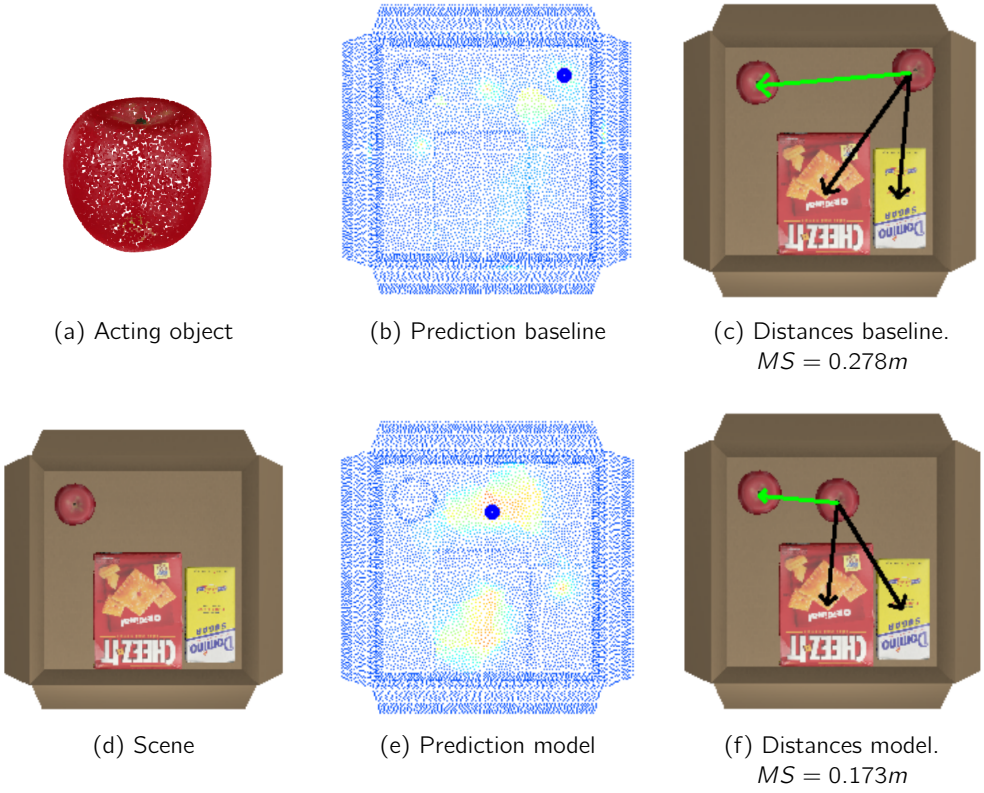
Figure 3.2: Comparison between baseline and developed model for a bleach cleanser. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with MS and MD distances visualized with a green and red arrow, respectively.

several other results for the same acting object are shown. In table 3.2 the calculated values are shown.

From these values it can be seen that the mean MS of the model is lower than the mean MD meaning that the model is successful in making semantically plausible predictions. At the same time, the mean MS is of the model is lower than that of the baseline model, together with the standard deviation. A lower standard deviation indicates a lower variance in the distances MS, meaning it performs repeatedly well. The baseline model predicts stable placement positions which can lie anywhere. Therefore, the MS and MD distances of the baseline model have a higher standard deviation, as observed. Furthermore, the mean MD of the model is higher than that of the baseline, indicating that it places the item further away from items belonging to another class. Based on the large difference between the baseline and the model, it can thus be concluded that the model is successful in predicting semantically stable positions for the bleach cleanser.

In figure 3.3 another comparison be seen. In this experiment, there are three items

in the box from which two are hard stackable items: sugar box and cracker box, and one is soft: apple. The acting object is an apple. As described in section 2.3, a soft item can be placed nearby other soft items or on top of hard stackable items. This means the predicted positions with a high value should lie on top of the two hard stackable items and around the apple.



(a) Acting object (b) Prediction baseline (c) Distances baseline. $MS = 0.278m$

(d) Scene (e) Prediction model (f) Distances model. $MS = 0.173m$

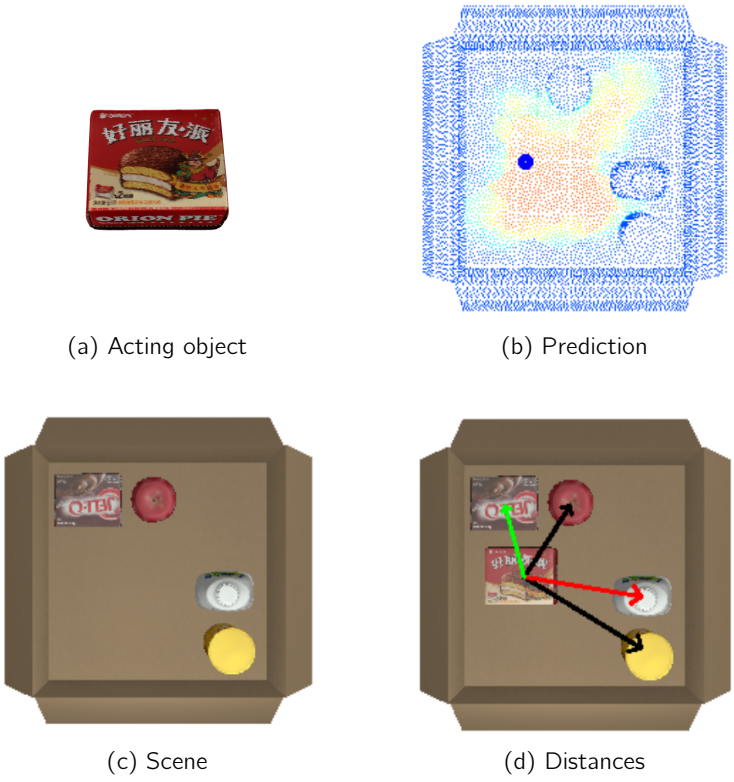
Figure 3.3: Comparison between baseline and developed model for an apple. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with the MS visualized with a green arrow.

In figure 3.3b the predictions of the baseline model are shown. It can be observed that the baseline model has difficulty in recognizing empty spaces as the predicted positions are colored in light yellow. The position with the highest predicted value, highlighted by a blue dot, is located at the top right away from the apple and two stackable items it can be placed on. In figure 3.3e the prediction of the model is shown and the model predicts regions located close to the apple and on top of the cracker box which is semantically plausible. The position with the highest predicted value is located next to the apple. In this case, the minimum distance to an object of a different class(MD) is invalid because the apple can be placed on top of the two hard stackable items. Thus to evaluate the performance, all of the scene objects are considered to be the same class. The minimum distance for the baseline is calculated to be 0.278m and for the model it is 0.173m. In appendix 4.2, figures 5 and 6 several other predictions are shown for the same acting object and scene in which the model successfully predicts that the apple can be placed

on top of hard stackable objects. It is observable that the model is successful in learning that a soft item like an apple can be placed on top of hard stackable items or nearby another soft apple.

3.4 Novel items

While the model is able to predict well for items that it has seen during training, it is interesting to observe what happens for a previously unseen item. To test whether the model generalizes to novel items, the following experiments are performed: for two previously unseen items being an orion pie and a cleanser, random scenes are generated and used to predict where they should be placed in the scene. In figure 3.4 the result is shown for orion pie.



(a) Acting object (b) Prediction

(c) Scene (d) Distances

Figure 3.4: Prediction for orion pie. The model takes as input the pointcloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.155$, $MD = 0.249$

	MS (in m)	MD (in m)
Figure 3.4	0.155	0.249
Mean	0.131	0.287
SD	0.023	0.042

Table 3.3: Calculated distances for orion pie

The scene contains four items in the box with two being soft: pudding box and apple and two are hard: bleach cleanser and chips can. The acting item to be placed is orion pie which the model has never seen before. Orion pie is a soft sensitive item. Therefore, it should be placed around the two other soft items. If the model is able to generalize well to unseen items, it is expected that the model has learned that a soft item like orion pie is more similar to the other soft items than the bleach cleanser and chips can and that it predicts positions located around the two soft items.

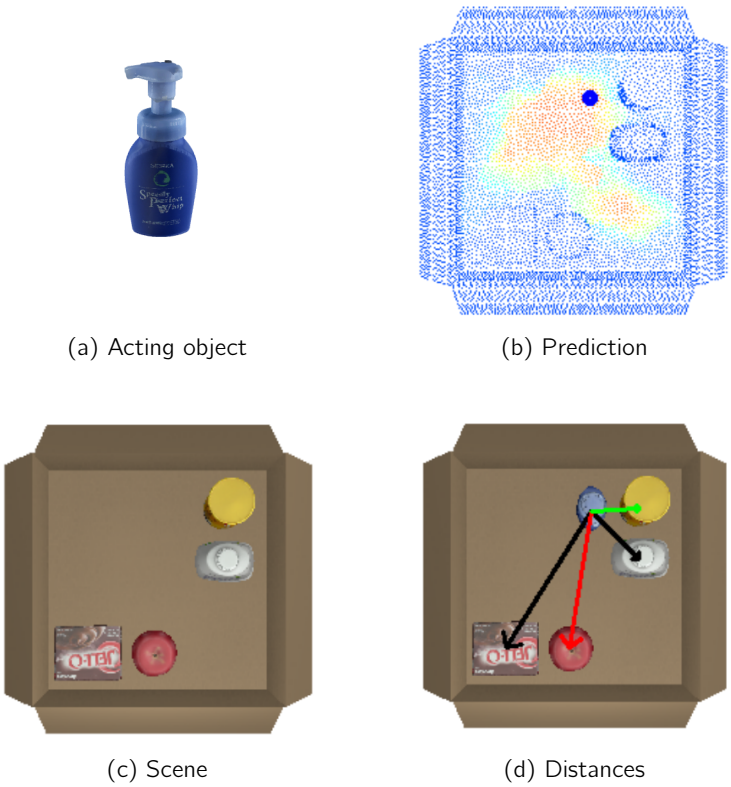
In figure 3.4b the prediction is shown. It can be observed that the predicted position marked with a blue dot is stable and lies around the two other soft away from the chips can and bleach cleanser. In figure 3.4d, the cleanser is placed on the predicted position and the distances to the scene objects are represented as arrows with MS and MD colored in green and red, respectively. In table 3.3 the calculated values for MS and MD are shown. The experiment is repeated ten times and the mean and standard deviations are computed. In appendix 4.2, figures 7 and 8 two other results are shown for the same acting object and similar scenes. The mean value for MS; μ_{MS} is significantly smaller than the mean value for MD; μ_{MD} meaning that the model is able to learn that a new item like orion pie possesses similar properties such as shape and color like the two other soft items and that it should be placed near them. The low standard deviations show that the model's predictions have low variance and thus, it is repeatedly successful in making plausible predictions.

Another novel item is a cleanser for which the model is also tested. In figure 3.5 the result is shown for a cleanser bottle.

	MS (in m)	MD (in m)
Figure 3.5	0.104	0.296
Mean	0.152	0.298
SD	0.053	0.01

Table 3.4: Calculated distances for cleanser

The scene contains the same items as in the previous case. The acting object to be placed is a cleanser which the model has never seen before. A bottle of cleanser is hard and non-stackable. Therefore, it should be placed around the two other hard items. If the model is able to generalize well to novel items, it is expected that the model has learned that the cleanser is more similar to the other hard items than the soft items and that it predicts positions located around the two soft items. In figure 3.5b the prediction is shown. It can be observed that red colored regions lie around the hard items and the predicted position lies closer to the hard items than to the two other soft items. In figure 3.5d the distances to the scene objects are represented as arrows with MS and MD colored in green and red, respectively. In table 3.4 the calculated values for MS and MD are shown. The experiment is repeated ten times and the mean and standard deviations are computed. In appendix 4.2, figures 9 and 10 two other results are shown for the same acting object and similar scenes. Just like the experiment with the orion pie, the mean value for MS; μ_{MS} is significantly smaller than the mean value for MD; μ_{MD}. From these mean values together with their low standard deviation, it can be concluded that the model is able to learn that a new item like a cleanser has similar



(a) Acting object (b) Prediction

(c) Scene (d) Distances

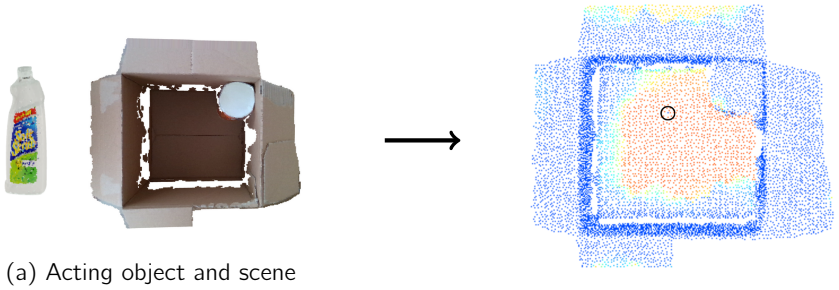
Figure 3.5: Prediction for cleanser. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.104m$, $MD = 0.296m$

properties such as shape and color like the two other hard items and that it should be placed near them.

3.5 Real data

To test whether the trained model generalizes well to real data, the model is evaluated on real data of boxes containing items. To capture real pointcloud data of boxes, an Intel RealSense d415 depth camera is used. A square carton box is used that resembles the one used during training, and various grocery store items are used, such as a chips can, window spray, oranges, and a cracker box. Several scenes are created by placing these items in the box. Using the Intel RealSense depth camera, a snapshot from above the box is taken, and a pointcloud containing position and color is obtained. Once the pointclouds are obtained, they are evaluated for various objects. In figure 3.6, 3.7, 3.8 three scenes are shown together with the predictions for the item to be placed. The positions with the highest predicted value are colored in blue and highlighted by black circle.

In figure 3.6a, the item to be placed is a bleach cleanser, and the item in the box is a chips can. Thus, the model should predict that the bleach cleanser must be placed

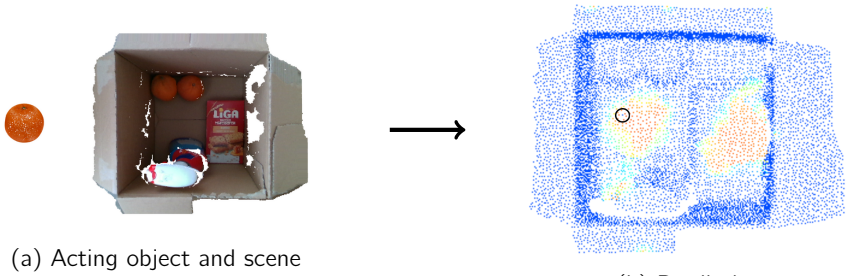


(a) Acting object and scene

(b) Prediction

Figure 3.6: Evaluation of a bleach cleanser on real scene containing a chips can. The model takes as input the pointcloud of the acting object and scene(a). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a black circle.

nearby the chips can as they are both hard items and nonstackable. In figure 3.6b the prediction made by the model is shown. It can be observed that red regions lie around the chips can, but they could have been less spread out. The highest predicted position lies nearby the chips can.



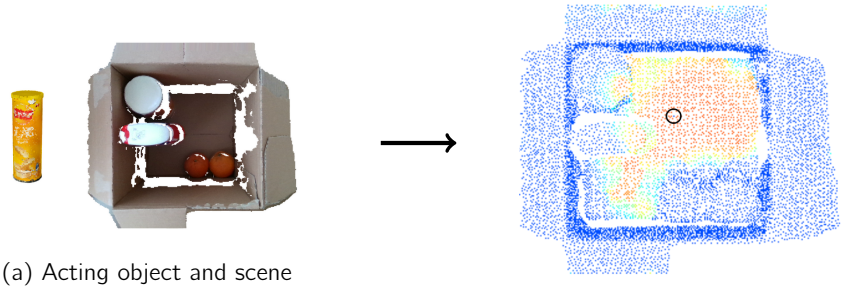
(a) Acting object and scene

(b) Prediction

Figure 3.7: Evaluation of an orange on a real scene containing two oranges, a cracker box and window spray. The model takes as input the pointcloud of the acting object and scene(a). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a black circle.

In figure 3.7, the item to be placed is an orange, and the box already contains oranges, a cracker box, and a window spray. Because the orange is soft, it should be placed near the other oranges or on top of the hard cracker box. In figure 3.7b, the prediction is shown, and it can be seen that the model is predicting the regions around the oranges and above the cracker box as suitable. However, there are also incorrectly red colored regions on top of the window spray bottle, which is unsuitable for placement. The position with the highest value lies nearby the oranges which is a semantically plausible stable position.

In figure 3.8, the item to be placed is a chips can, and the box contains oranges, a chips can, and a window spray. The chips can must be predicted to be placed nearby the chips can and window spray, and in figure 3.8b, it can be seen that the predicted position lies next to the chips can and window spray. However, in this case, the model is also having difficulty as the red colored regions could have been less spread out.



(a) Acting object and scene

(b) Prediction

Figure 3.8: Evaluation of a chips can on a real scene containing two oranges, a chips can and window spray. The model takes as input the pointcloud of the acting object and scene(a). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a black circle.

From these examples, it can be seen that the model is able to learn the relationships between objects, and it is able to generalize to unseen real pointcloud data of boxes. However, the model is also having difficulty as the prediced regions on the heatmaps could have been less spread out, and some predicted regions laid on top of other unsuitable items. The model having difficulty can be explained by a few factors: First of all, the box used to obtain real data is not the same as the one the model has seen during training in the simulator. Secondly, the real items used to generate real box configurations are not seen before by the model during training. While they are similar, they are not the same. Lastly, there are varying environmental conditions. The real data was obtained during the daytime without the optimal lighting conditions that were occurrent in the simulator. At the same time, the snapshot is taken from above at a fixed position in the simulator. In the real case, the snapshot is taken by holding the camera above the box using my hand which led to the varying angles compared to the simulator.

3.6 Packing a list of items

While the previous sections show that the model is successful in packing a single item in an arbitrary scene, the trained model is also tested to evaluate whether a list of items can be packed according to their predicted positions. To test the model, the following experiments are performed: an initial empty box is taken, and an arbitrary list of acting items to be placed is generated. The model predicts a position for each acting item on which the item is placed. The model is tested to pack a various number of items and in figure 3.9, the result is shown for packing three items in a box.

The initial box is empty and the first item to be placed is a gelatin box which should be placed in one of the corners. In the second image from top left it is shown that the model has predicted a corner as a placement position and in the third image the placement is shown. The next item to be placed is a chips can, which should be placed away from the gelatin box. In the top right image it is observable that the model has learned this and predicts the opposite corner. The third item to be placed is a bleach

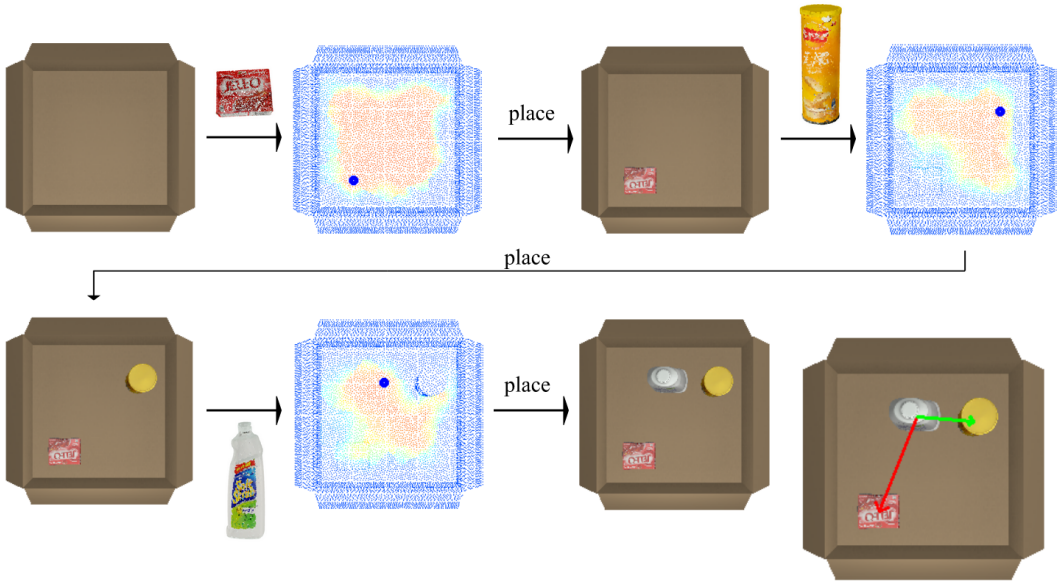


Figure 3.9: Packing three items using model. For each item to be packed, the predicted heatmap is shown with the position with the highest predicted value highlighted with a blue dot. The MS and MD distances are represented with green and red arrows in bottom right image, respectively.

cleanser which should be placed nearby the chips can. In the second image from the bottom left, it can be seen that the model predicts to place the bleach cleanser nearby the chips can. In the bottom right image the distances MS and MD are represented as a green and red arrow, respectively. The calculated value for MS is found to be 0.153m and the MD 0.274m. From the observed placement and calculated distances it is observable that the MS is significantly smaller than the MD. From this example, it is observable that the model can successfully learn the semantic relationships between items and place the three items accordingly.

The model is also tested on packing more items and in figure 3.10 the result is shown for packing seven items in the following order: sugar box, apple, bleach cleanser, pudding box, chips can, peach and orange, respectively. There are three items that are hard: the sugar box, bleach cleanser and chips can. The rest of the items are soft. For each of the items, the model should predict a position where the MS is smaller than the MD. In the figure, the predicted heatmaps are shown in which the position with the highest predicted value is highlighted with a dark blue dot. In the figure, also images are shown where the object is placed on the predicted position together with the distances to other scene objects represented as arrows. The MS and MD distances are represented with a green and red arrow, respectively. For each acting object where the MS and MD are applicable, the values are calculated and shown in table 3.5. By looking at the table, it can be observed that the MS is smaller than the MD for each acting object. The mean MS over all placed items is also smaller than the mean MD meaning that the model is successful in packing seven items on plausible positions. It has learned the semantic relationships between items and places similar items closer together on stable positions.

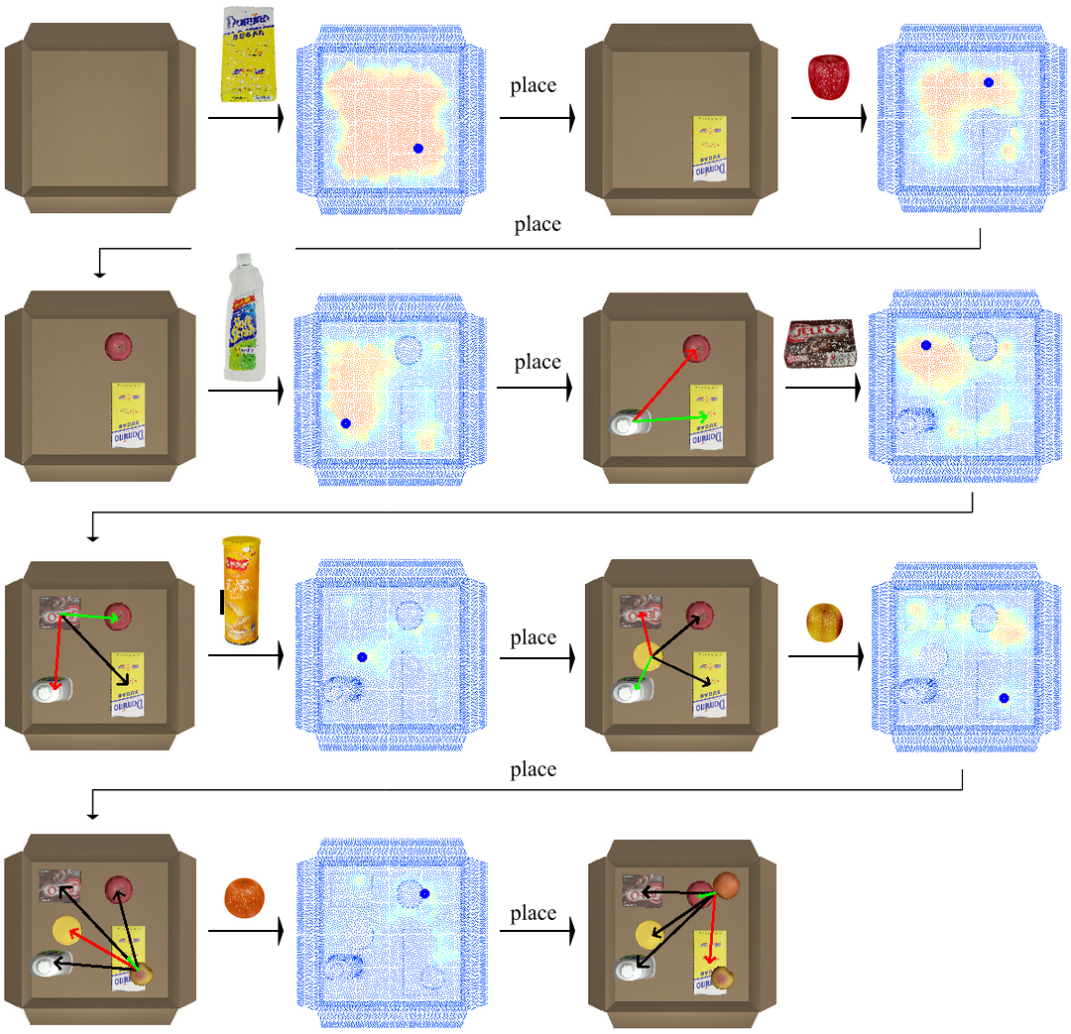


Figure 3.10: Packing seven items using model. For each item to be packed, the predicted heatmap is shown with the position with the highest predicted value highlighted with a blue dot. The MS and MD distances, where applicable, are represented with green and red arrows, respectively.

Acting object	*MS* (in m)	*MD* (in m)
Bleach cleanser	0.249	0.321
Pudding box	0.193	0.258
Chips can	0.121	0.153
Peach	0.053	n.a.
Orange	0.057	0.240
Mean	0.155	0.243

Table 3.5: Calculated distances MS and MD during placement using developed model for the case in figure 3.10

The same order of items is also packed using the baseline model. In figure 3.11, the results for the baseline model are shown. It can be observed that the baseline model predicts empty spots lying around the middle for the first three items. However, the baseline model has difficulty in recognizing empty spaces which can be seen by the lack

of red colored regions on the prediction heatmap. One item, the pudding box is even predicted to be placed on the edge of the box resulting in the pudding box falling down inside the box. When looking at the calculated distances for MS and MD using the baseline model in table 3.6, it can be observed that each of the items is placed closer to an item of a different class than an item of the same class; the MS is higher than MD for each item. This can also be concluded from the mean MS which is significantly higher than the mean MD.

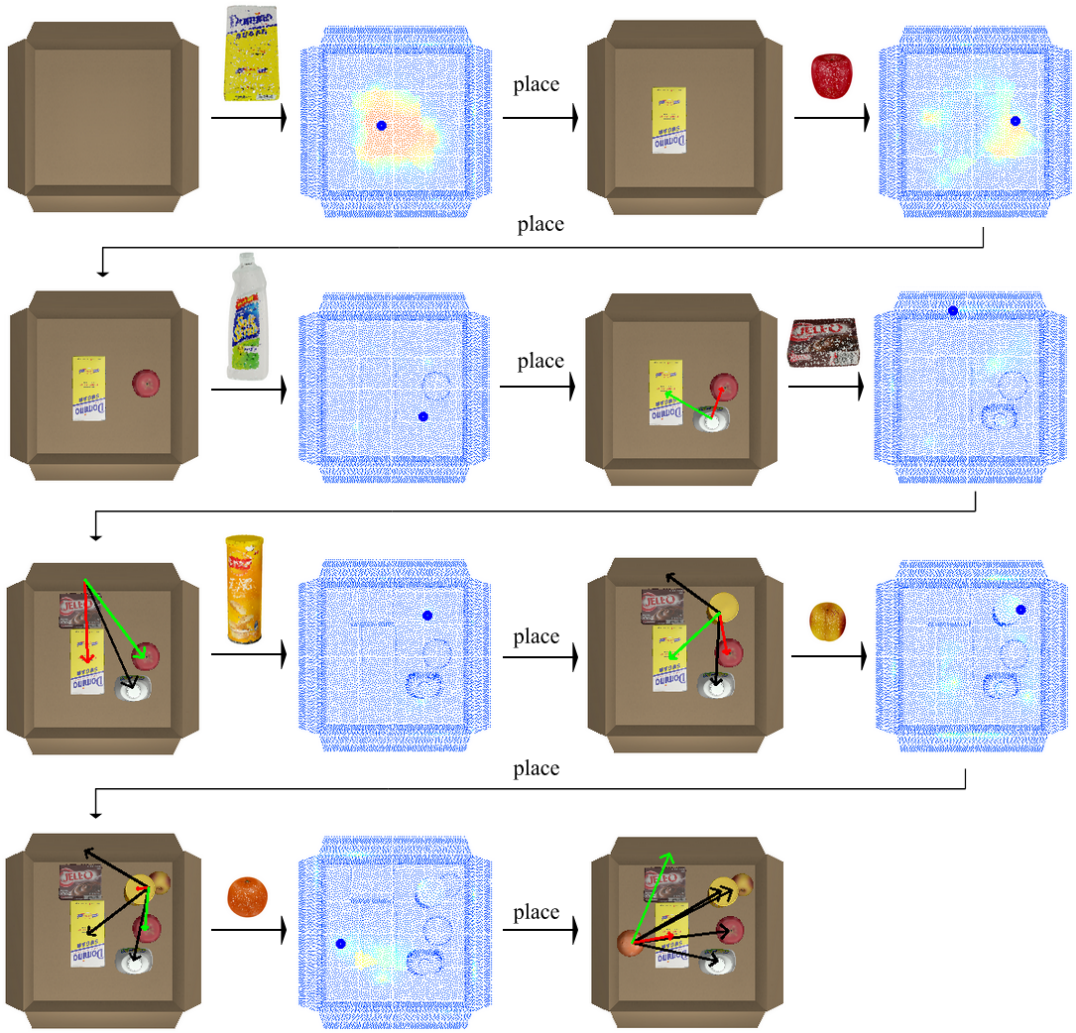


Figure 3.11: Packing seven items using baseline model. For each item to be packed, the predicted heatmap is shown with the position with the highest predicted value highlighted with a blue dot. The MS and MD distances, where applicable, are represented with green and red arrows, respectively.

When comparing the final scene with the developed model, it can be observed that all of the items lie around the middle without soft and hard items being separated. The developed model, however, is able to learn these relationships between objects and place items stably accordingly.

The MS and MD distances are also calculated after placement of all items. Rather than looking only at the distances during placement, the MS and MD distances are also calculated for each item after placement of all items to see whether items belonging to

Acting object	*MS* (in m)	*MD* (in m)
Bleach cleanser	0.175	0.109
Pudding box	0.296	0.241
Chips can	0.224	0.139
Peach	0.140	0.036
Orange	0.298	0.139
Mean	0.227	0.133

Table 3.6: Calculated distances MS and MD during placement using baseline model for the case in 3.11

the same class lie closer to each other than to objects of a different class at the end. In table 3.7, two tables are shown containing the calculated MS and MD distances after placement of all items. It can be observed the mean MS for the developed model is lower than the mean MD in table 3.7b. This shows that using the developed model, the items are placed closer to items belonging to the same class resulting in a scene in which items of the same class lie closer to each other than to items of a different class. When comparing the developed model with the baseline, the mean MS for the developed model is significantly lower than the mean MS for the baseline. The MS of the baseline is also higher than the MD indicating that the baseline results in placements in which items belonging to the same class do not lie closer to eachother than to items belonging to a different class. These comparisons show that the developed model is able to learn the relationships between objects and place items stably accordingly; items belonging to the same class lie closer together and away from items belonging to a different class. However, using the baseline, the items lie around the middle without soft and hard items being separated.

Acting object	MS (in m)	MD (in m)
Sugar box	0.175	0.139
Apple	0.140	0.110
Bleach cleanser	0.175	0.110
Pudding box	0.220	0.189
Chips can	0.224	0.036
Peach	0.140	0.036
Orange	0.298	0.139
Mean	0.196	0.108

(a) Baseline

Acting object	MS (in m)	MD (in m)
Sugar box	0.217	0.230
Apple	0.057	0.207
Bleach cleanser	0.121	0.258
Pudding box	0.193	0.153
Chips can	0.121	0.153
Peach	0.053	n.a.
Orange	0.057	0.240
Mean	0.127	0.207

(b) Model

Table 3.7: Calculated distances MS and MD after placement for the case in figures 3.10 and 3.11

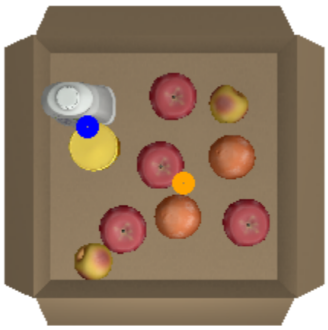
The baseline and developed model are also tested on a more difficult scenario where there are ten items to be packed of which eight are soft and two are hard. The initial scene is empty and the order of items (from left to right) to be placed is shown in figure 3.12b. The bleach cleanser and chips can are hard, all other items are soft. In figures 3.12c and 3.12d the scenes after packing all ten items is visualized using the baseline and developed model, respectively. In the figures the cluster center of hard items is visualized with a blue dot and the cluster center for soft items with an orange dot.

The cluster centers are computed by taking the average position of the hard items and soft items. It can be observed that using the baseline model, the soft and hard items are placed nearby each other. The items are spread over the surface with some items lying closer to objects of a different class. In figure 3.12d it can be seen that using the developed model, the soft and hard items are placed separately and nearby items belonging to the same class. The cluster centers of hard and soft items are further apart using the developed model. This can also be concluded from table 3.8 in which the calculated distances MS and MD are shown after placement of all items. It can be observed that for the developed model, all placed items lie closer to items belonging to the same class than to items of a different class; the MS is smaller than MD for all items. The mean MS for the developed model is also significantly lower than the mean MD. This is not the case for the baseline model. The baseline model results in a scene in which some items are placed closer to objects belonging to a different class. While the mean MS for the baseline model is smaller than the mean MD, the difference is not much compared to the developed model.

(a) Initial scene

(b) List of items

(c) Resulting scene using baseline



(d) Resulting scene using model

Figure 3.12: Packing ten items using baseline and developed model. The ten items (b) are packed one by one in the initial box (a) using the baseline and developed model. The resulting scene after packing all ten items are shown (c and d) in which the cluster center for soft items is represented with an orange dot and the cluster center for hard items with a blue dot.

Acting object	MS (in m)	MD (in m)
Apple	0.140	0.303
Orange	0.053	0.215
Apple	0.178	0.106
Bleach cleanser	0.156	0.106
Apple	0.142	0.125
Apple	0.053	0.221
Orange	0.181	0.082
Chips can	0.156	0.106
Peach	0.196	0.106
Peach	0.052	0.220
Mean	0.131	0.159

(a) Baseline

Acting object	MS (in m)	MD (in m)
Apple	0.115	0.199
Orange	0.122	0.271
Apple	0.053	0.167
Bleach cleanser	0.106	0.199
Apple	0.111	0.122
Apple	0.126	0.325
Orange	0.111	0.211
Chips can	0.106	0.122
Peach	0.115	0.275
Peach	0.053	0.212
Mean	0.102	0.210

(b) Model

Table 3.8: Calculated distances MS and MD after placement of all items for the case in figure 3.12

4. Discussion and Conclusion

4.1 Discussion

This thesis aimed to learn object-object affordances to predict where an item should be placed in a box during an item packing task. Using a simulator called SAPIEN and objects from the YCB and ShapeNet dataset, large-scale interaction data simulating item packing is generated. After generating large-scale realistic interaction data where items are placed in stable and semantically plausible positions, i.e., placing items according to their categorical labels, the developed model is trained. Using the developed model, predictions are made where each point on the scene pointcloud is labeled with an affordance value with a higher value indicating a better placement. The position with the highest predicted value is then taken to be the predicted placement position. Hereby, the predicted positions must be both geometrically feasible and semantically plausible. By building an item packing pipeline that places an item on the predicted placement position, several experiments are performed to analyze the model's performance, compare it with the baseline model and evaluate the generalizability of the model to novel items and real data. The predictions made by the model are evaluated empirically by observing the heatmaps and quantitatively using the minimum distance to object of same class (MS) and minimum distance to object of a different class (MD) metrics for the predicted position.

Results show that the model is successful in making predictions that are both stable and semantically plausible, following the learned relationships between items it has seen during training. The predicted positions where the items are placed lie closer to items belonging to the same class and further away from items belonging to another class. The model is also successful in learning cases where soft items can be placed on top of hard stackable items or nearby similar items. The MS being smaller than MD and the low standard deviations of the predictions indicate that the model successfully makes semantically plausible stable predictions. While the predicted positions are plausible, the predicted heatmaps could have been less spread out; red regions could have been clustered closer to items it belongs to. Comparisons with the baseline model show a significant difference in the predictions. While the baseline model is able to predict stable positions, the model has learned the semantic relationships and places items stably accordingly.

To evaluate whether the model can generalize to novel items, experiments with two novel items are carried out. For two novel items, scenes are generated and using the model, predictions are made repeatedly. Results show that predictions made by the model lie closer to the items with similar properties than those without; the MS is smaller than the MD. Based on the computed mean values for MS and MD and standard deviations, it is found that the model is repeatedly making plausible predictions.

Thus, the model is successful in adapting to novel items.

To evaluate whether the model can generalize to real data, an Intel RealSense depth camera is used to capture real pointcloud data of boxes. For several known items, predictions are made on real world data. It can be observed that the predicted positions lie closer to items belonging to the same class. However, the model is having difficulty as the predicted heatmaps are more spread out than they could have been. There are also red-colored regions unsuitably located on top of other items, which is undesired. The reason for the model having difficulty on real pointcloud data of boxes is that the used box and items inside the box are both novel for the model. While a similar box and items are used during data generation in the simulator, they are not the same. At the same time, there is a difference in the environmental conditions between the simulator and the real world. Therefore, the model's performance on real data is believed to improve if real data is combined with the generated synthetic data during the training of the model. Using this method, the training data can be enriched, and the model can learn properties intrinsic to real data.

Experiments have also been performed in packing a list of items as opposed to single items. In section 3.6, several item packing experiments are performed in which a list of items in a random order is packed. The model has been tested on packing three, seven, and ten items, respectively. It can be observed that the model's predictions result in the successful packing of items. The items are predicted and placed on positions that lie closer to items belonging to the same class; The MS is smaller than MD. It can also be observed that the model has learned that soft items can be placed near other soft items and on top of hard stackable items. Comparisons with the baseline show that the baseline results in placements in which soft and hard items are placed randomly on stable positions. However, the developed model results in placements where items belonging to the same class are grouped together away from items belonging to a different class.

4.2 Conclusion

The aim of this thesis was to learn object-object affordances to predict a placement position in a box in an item packing task. Using a simulator called SAPIEN and objects from the YCB and ShapeNet dataset, large-scale interaction data simulating item packing is generated. After generating large-scale interaction data where items are placed in stable positions and on semantically plausible positions, i.e., placing items according to their categorical given labels, a developed model is trained. The model takes as input a complete 6D pointcloud of the item to be placed and a partial 6D pointcloud of the box in which each point consists of XYZ coordinates and RGB colors. The output of the model is an affordance heatmap where each point on the scene pointcloud is labeled with a value between zero and one, with a higher value indicating a better placement. The position with the highest predicted value is then taken to be the predicted placement position. Thus, the predicted positions must be both geometrically feasible and semantically plausible. By building an item packing pipeline that uses the trained model to pack items, experiments are performed to evaluate the performance on packing single items, novel items, real data, and packing a list of items.

Results show that the model is successful in making predictions that are both stable and semantically plausible. The predicted positions where the items are placed lie closer to items belonging to the same class and further away from items belonging to another class. The model has also successfully learned cases where soft items can be placed on top of hard stackable items or nearby similar items.

Future work could be done to investigate how the predictions change with more items and/or different boxes. For example, more items and/or different boxes can be used rather than the selected fourteen items and a single box. At the same time, exploring a variable orientation of the items can be interesting as they are kept fixed now.

Considering the performance on real data, another future work could be done by adding real data to the training data set and investigating how the predictions improve. Furthermore, it could also be explored how to deploy the built pipeline on a real robot. Because of the usage of depth cameras like Intel RealSense, deployment on a real robot is suitable.

Another interesting further research could be done in developing and incorporating a cost function that maximizes the empty free space during placements. This could help to decrease the distance between placed objects to pack the items more tightly together and maximize the free space during item packing.

Bibliography

[1] Barbara Baarsma and Jesse Groenewegen. "COVID-19 and the Demand for Online Grocery Shopping: Empirical Evidence from the Netherlands". In: *De Economist* 169 (July 2021). doi: 10.1007/s10645-021-09389-y.

[2] Dylan Jennings and Miguel Figliozzi. "Study of Road Autonomous Delivery Robots and Their Potential Effects on Freight Efficiency and Travel". In: *Transportation Research Record* 2674.9 (2020), pp. 1019–1029. doi: 10.1177/0361198120933633. eprint: https://doi.org/10.1177/0361198120933633. url: https://doi.org/10.1177/0361198120933633.

[3] Lionel Birglen. "Design of a partially-coupled self-adaptive robotic finger optimized for collaborative robots". In: *Autonomous Robots* 43 (Feb. 2019). doi: 10.1007/s10514-018-9802-x.

[4] Rahul Shome et al. *Tight Robot Packing in the Real World: A Complete Manipulation Pipeline with Robust Primitives*. 2019. doi: 10.48550/ARXIV.1903.00984. url: https://arxiv.org/abs/1903.00984.

[5] James J Gibson. "The theory of affordances". In: *Hilldale, USA* 1.2 (1977), pp. 67–82.

[6] Ian Lenz, Honglak Lee, and Ashutosh Saxena. *Deep Learning for Detecting Robotic Grasps*. 2014. arXiv: 1301.3592 [cs.LG].

[7] Yuzhe Qin et al. *S4G: Amodal Single-view Single-Shot SE(3) Grasp Detection in Cluttered Scenes*. 2019. arXiv: 1910.14218 [cs.RO].

[8] Joseph Redmon and Anelia Angelova. *Real-Time Grasp Detection Using Convolutional Neural Networks*. 2015. arXiv: 1412.3128 [cs.RO].

[9] Priyanka Mandikal and Kristen Grauman. *Learning Dexterous Grasping with Object-Centric Visual Affordances*. 2021. arXiv: 2009.01439 [cs.RO].

[10] Lixin Yang et al. *CPF: Learning a Contact Potential Field to Model the Hand-Object Interaction*. 2021. arXiv: 2012.00924 [cs.CV].

[11] Enric Corona et al. "GanHand: Predicting Human Grasp Affordances in Multi-Object Scenes". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5030–5040. doi: 10.1109/CVPR42600.2020.00508.

[12] Kuan Fang et al. "Demo2Vec: Reasoning Object Affordances from Online Videos". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2139–2147. doi: 10.1109/CVPR.2018.00228.

[13] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. *Grounded Human-Object Interaction Hotspots from Video*. 2019. arXiv: 1812.04558 [cs.CV].

[14] Kaichun Mo et al. *O2O-Afford: Annotation-Free Large-Scale Object-Object Affordance Learning*. 2021. arXiv: 2106.15087 [cs.CV].

[15] Yun Jiang et al. *Learning to Place New Objects in a Scene*. 2012. arXiv: 1202.1694 [cs.RO].

[16] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. doi: 10.48550/ARXIV.1512.03012. url: https://arxiv.org/abs/1512.03012.

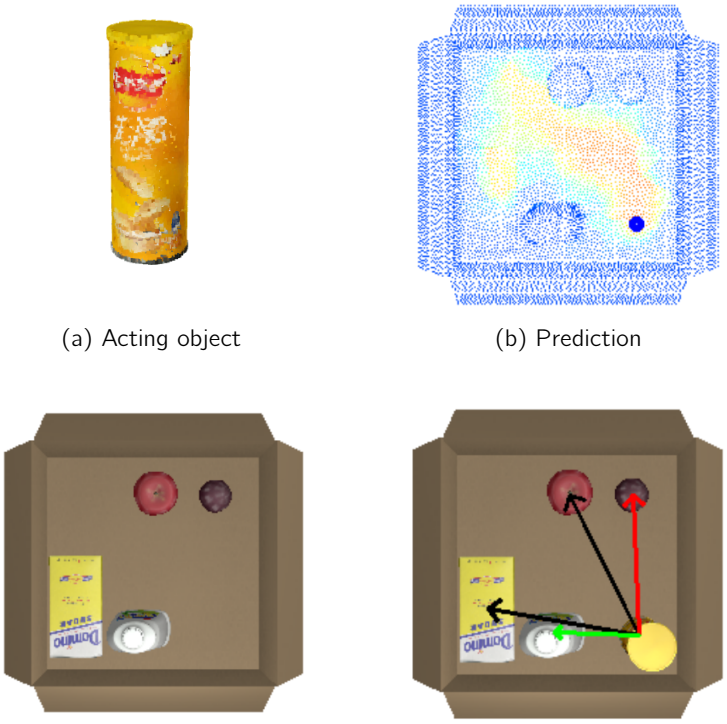
[17] Berk Calli et al. "The YCB object and Model set: Towards common benchmarks for manipulation research". In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015, pp. 510–517. doi: 10.1109/ICAR.2015.7251504.

[18] Fanbo Xiang et al. *SAPIEN: A SimulAted Part-based Interactive ENvironment*. 2020. doi: 10.48550/ARXIV.2003.08515. url: https://arxiv.org/abs/2003.08515.

[19] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. doi: 10.48550/ARXIV.1706.02413. url: https://arxiv.org/abs/1706.02413.

[20] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2016. doi: 10.48550/ARXIV.1612.00593. url: https://arxiv.org/abs/1612.00593.

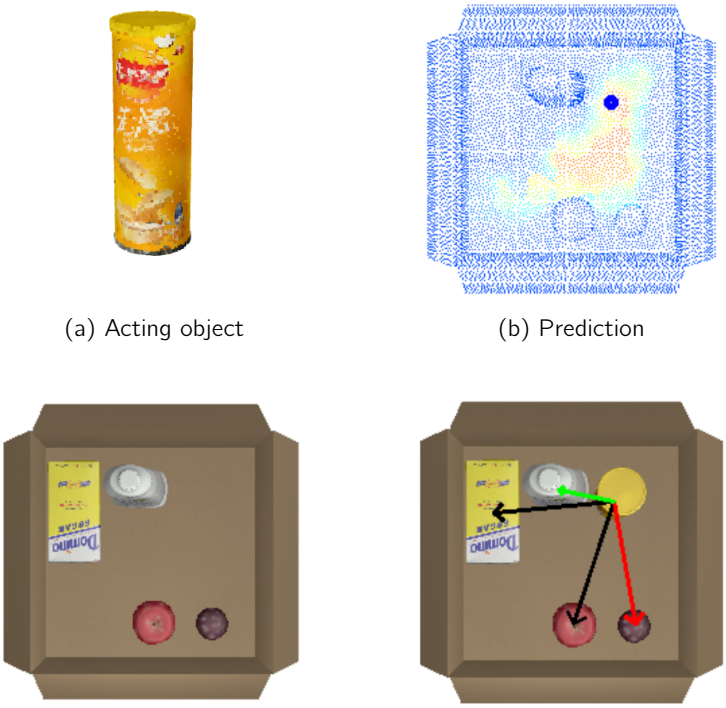
Appendix A



(a) Acting object (b) Prediction

(c) Scene (d) Distances

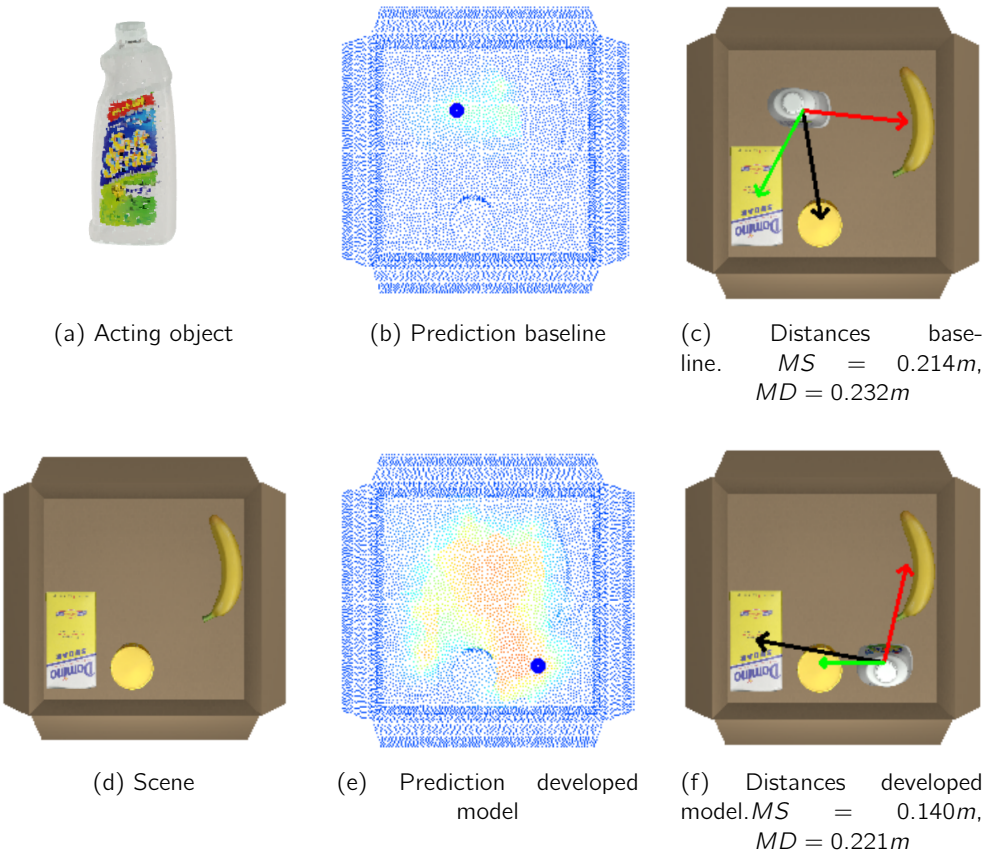
Figure 1: Prediction for chips can. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.176m$, $MD = 0.296m$



(a) Acting object (b) Prediction

(c) Scene (d) Distances

Figure 2: Prediction for chips can. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.120m$, $MD = 0.269m$



(a) Acting object

(b) Prediction baseline

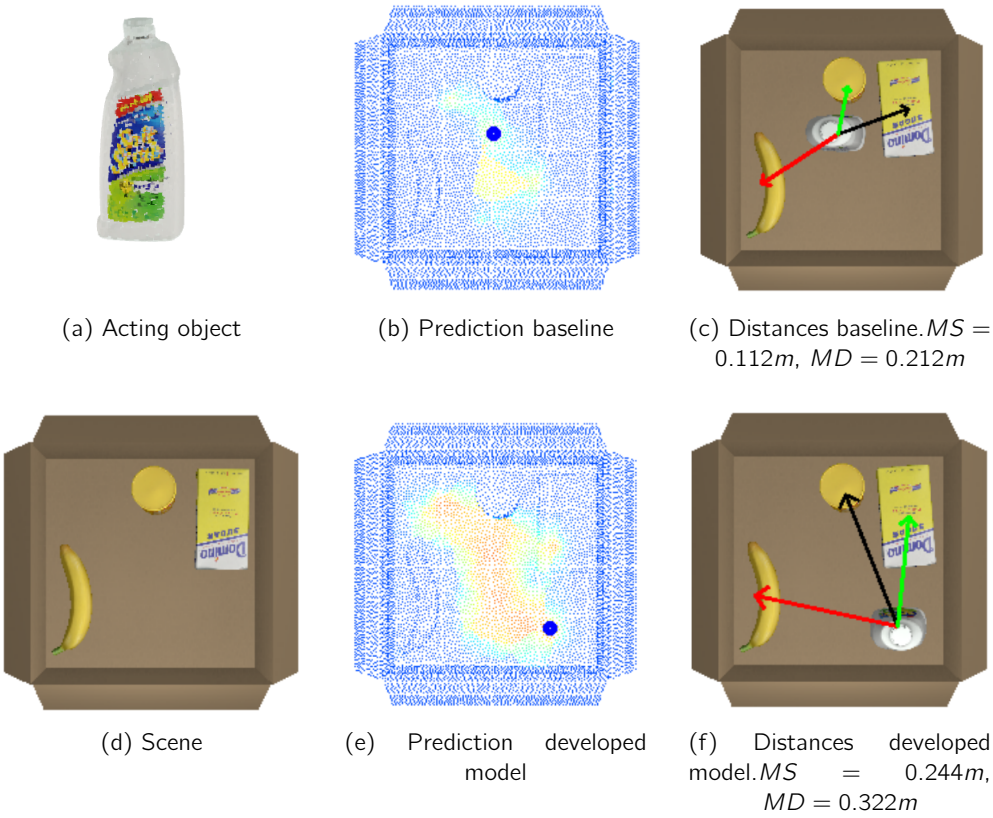
(c) Distances baseline. $MS = 0.214m$, $MD = 0.232m$

(d) Scene

(e) Prediction developed model

(f) Distances developed model. $MS = 0.140m$, $MD = 0.221m$

Figure 3: Comparison between baseline and developed model for a bleach cleanser. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with MS and MD distances visualized with a green and red arrow, respectively.



(a) Acting object

(b) Prediction baseline

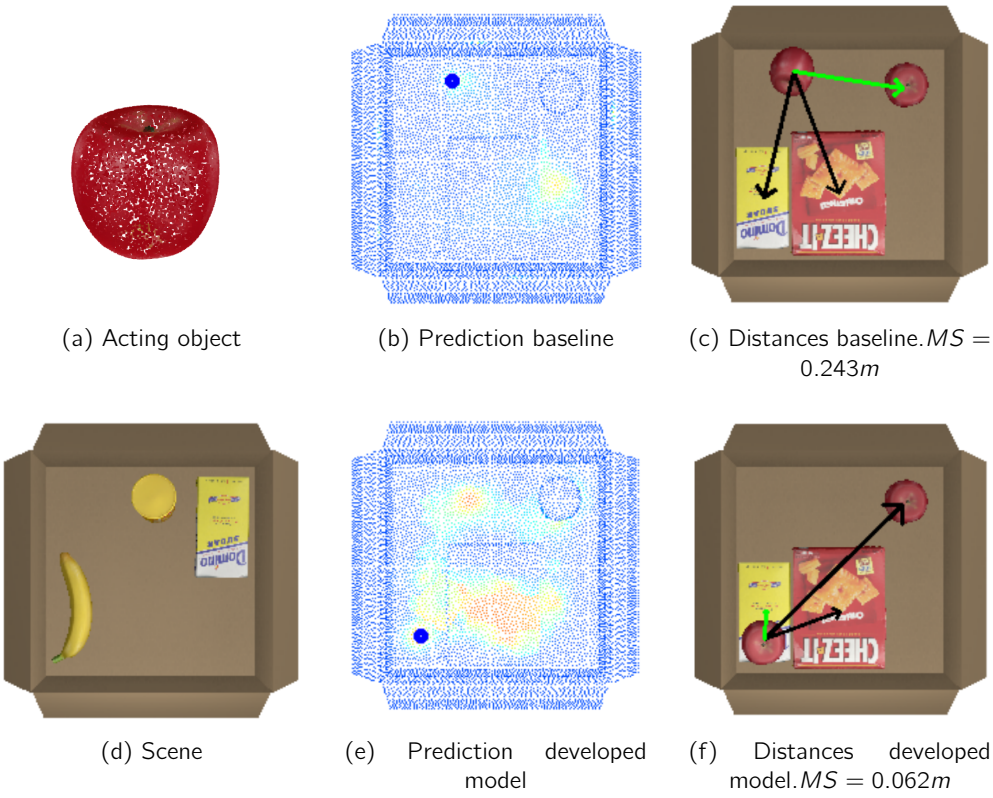
(c) Distances baseline. $MS = 0.112m$, $MD = 0.212m$

(d) Scene

(e) Prediction developed model

(f) Distances developed model. $MS = 0.244m$, $MD = 0.322m$

Figure 4: Comparison between baseline and developed model for a bleach cleanser. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with MS and MD distances visualized with a green and red arrow, respectively.



(a) Acting object

(b) Prediction baseline

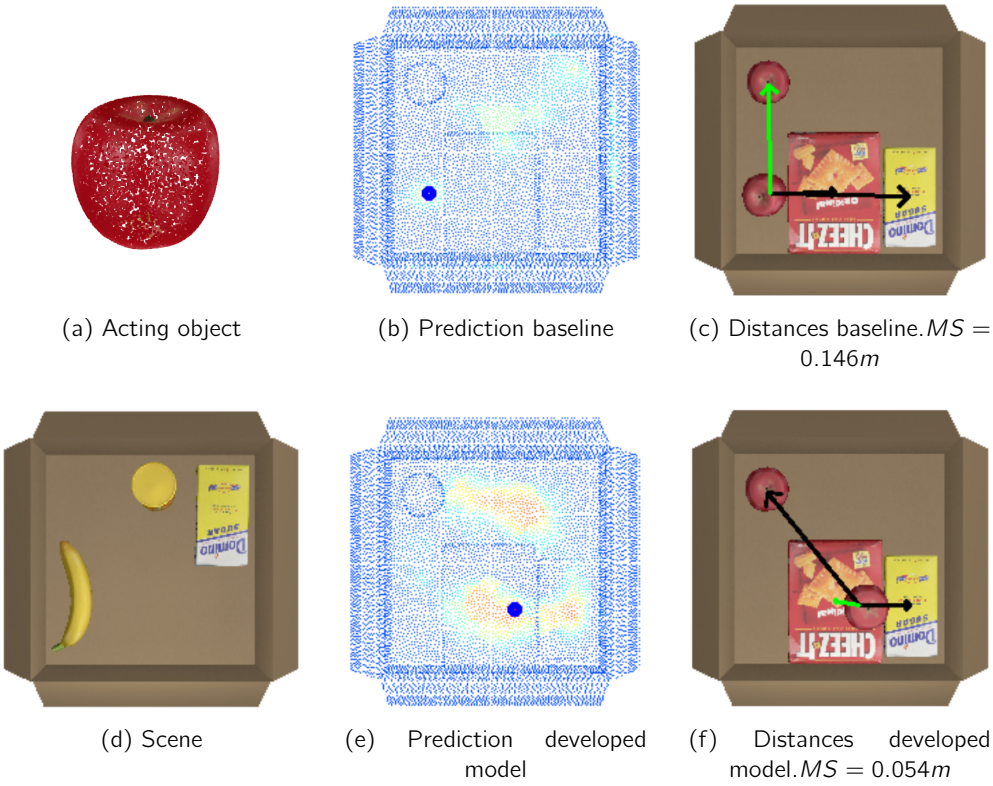
(c) Distances baseline. $MS = 0.243m$

(d) Scene

(e) Prediction developed model

(f) Distances developed model. $MS = 0.062m$

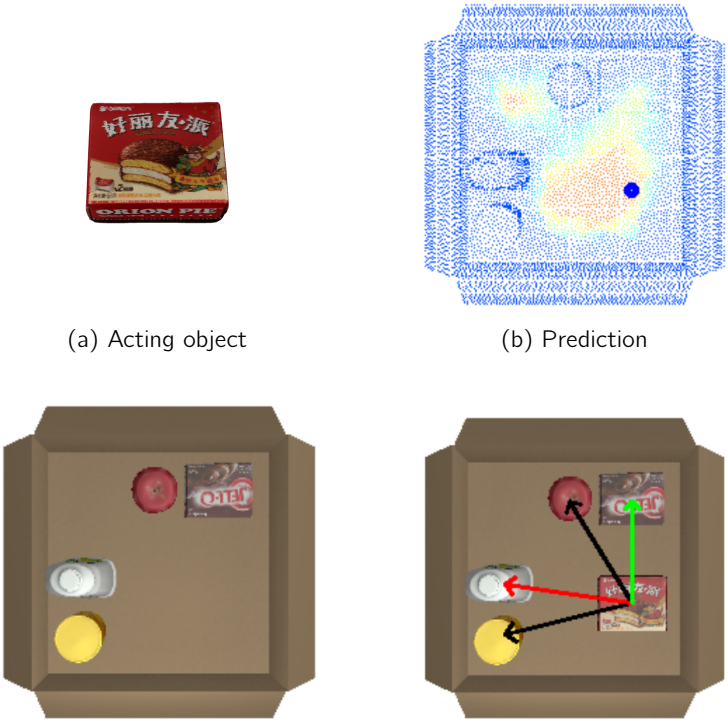
Figure 5: Comparison between baseline and developed model for an apple. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with MS distance visualized with a green arrow.



(a) Acting object (b) Prediction baseline (c) Distances baseline. $MS = 0.146m$

(d) Scene (e) Prediction developed model (f) Distances developed model. $MS = 0.054m$

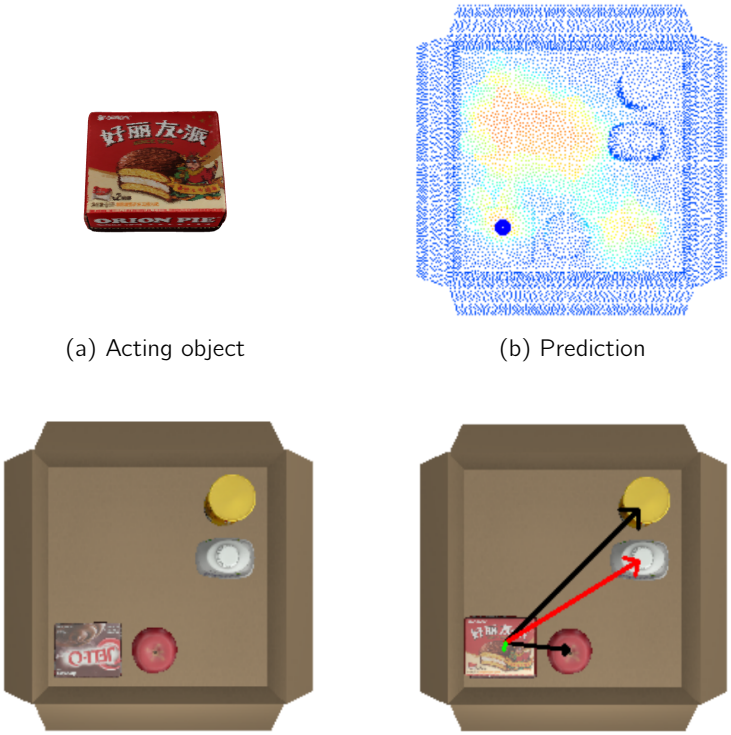
Figure 6: Comparison between baseline and developed model for an apple. The model takes as input the pointcloud of the acting object(a) and scene(d). The baseline and developed model output a prediction heatmap(b and e) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(c and f) with MS distance visualized with a green arrow.



(a) Acting object (b) Prediction

(c) Scene (d) Distances

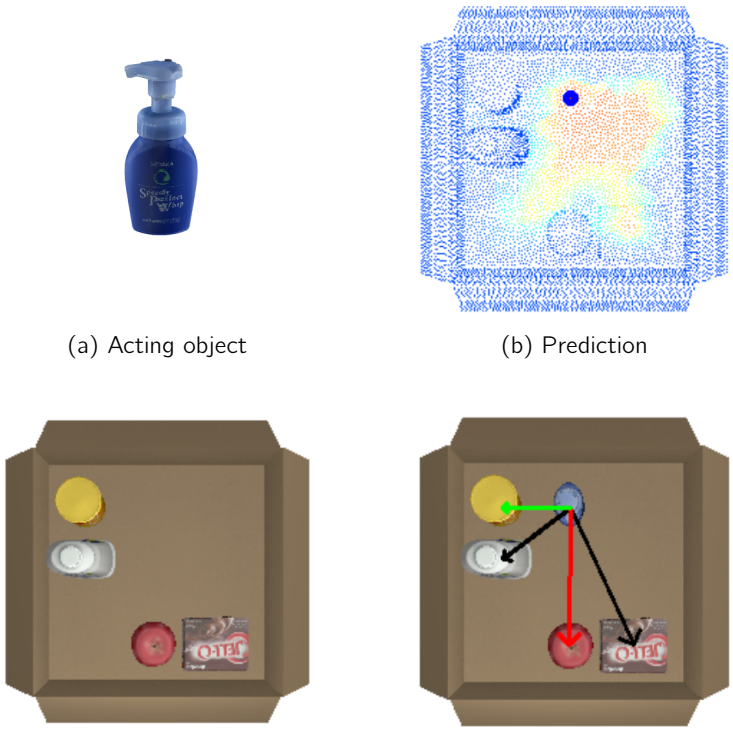
Figure 7: Prediction for orion pie. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.221m$, $MD = 0.280m$



(a) Acting object (b) Prediction

(c) Scene (d) Distances

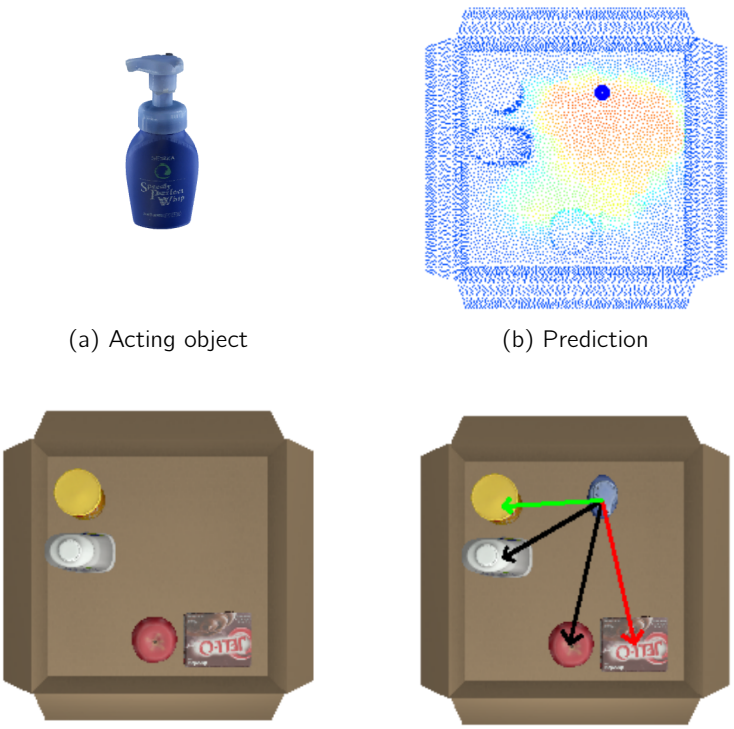
Figure 8: Prediction for orion pie. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.016m$, $MD = 0.333m$



(a) Acting object (b) Prediction

(c) Scene (d) Distances

Figure 9: Prediction for cleanser. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.143m$, $MD = 0.290m$



(a) Acting object (b) Prediction

(c) Scene (d) Distances

Figure 10: Prediction for cleanser. The model takes as input the point-cloud of the acting object(a) and scene(c). The model outputs a prediction heatmap(b) in which the position with the highest predicted value is highlighted with a blue dot. The item is then placed on that position resulting in scene(d) with MS and MD distances visualized with a green and red arrow, respectively. $MS = 0.209m$, $MD = 0.309m$

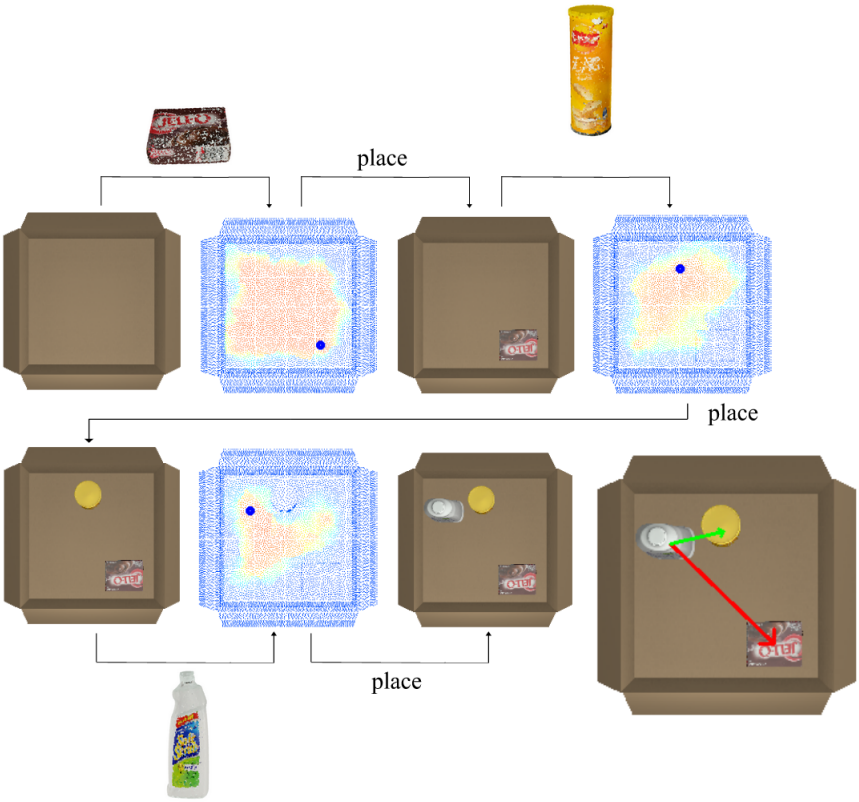


Figure 11: Packing 3 items. For each item to be packed, the predicted heatmap is shown with the position with the highest predicted value highlighted with a blue dot. The MS and MD distances are represented with green and red arrows in the last image, respectively. $MS = 0.137m$, $MD = 0.362m$