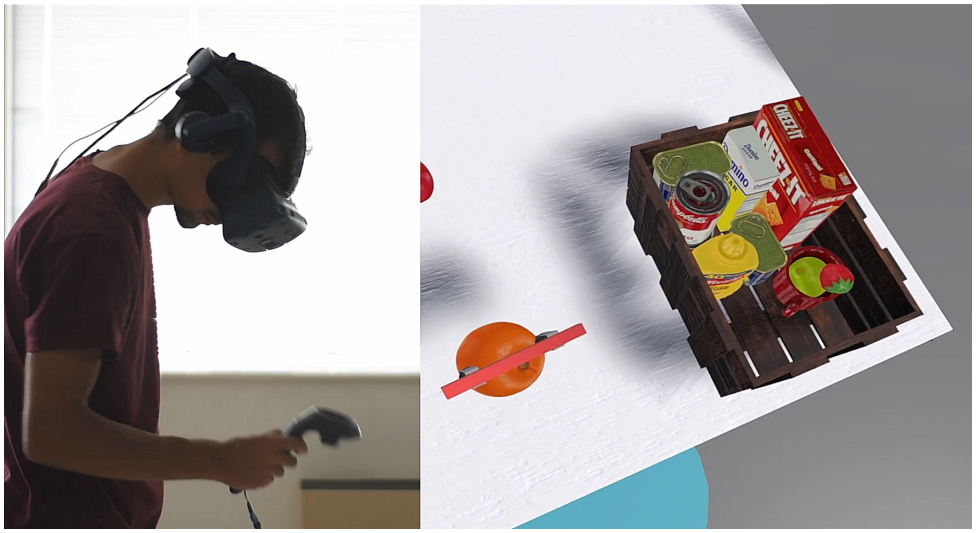
TÉCNICO LISBOA



Learning the Sequence of Packing Irregular Objects from Human Demonstrations

André José Freitas Santos

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Atabak Dehban
Prof. José Alberto Rosado dos Santos Victor

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Dr. Atabak Dehban
Member of the Committee: Prof. Manuel Fernando Cabido Peres Lopes

November 2022

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost, a genuine thank you to both my supervisors, Professor José Santos-Victor and Atabak Dehban. This work would not have been possible without you.

To Professor José Santos-Victor, thank you for the contagious desire to do more and better. In each meeting with you, your encouragement was an instant recharge of my will to expand and improve this work.

To Atabak Dehban, thank you for your unceasing availability. You were there at each step to help and guide me through all the difficulties and problems, both large and small.

And to both and above all else, thank you for your friendship. It was much more than I could have expected and I will never forget it.

To all the Vislab members, thank you for welcoming me so well into such an exciting group of people.

To Leonor, just thank you. Words will never be enough to describe my gratitude for everything you have done. You were there every second of every day, and I would not have achieved what I have today if it were not for you.

To Diogo, much more than a friend, thank you for all the priceless, wonderful moments of the last five years.

To all my closest friends, thank you for all the happy moments, which powered me during the last five years.

To my family, thank you for the endless support.

Abstract

Exclusively online supermarkets have been expanding in recent years. Their operation involves packing and shipping millions of orders as efficiently as possible. The automation of the former task, bin packing, has seen slow progress mainly due to the inherent complexity of safely packing irregular objects such as groceries. This task's underlying constraints on object placement and manipulation, and the diverse objects' physical properties make preprogrammed strategies unfeasible.

Our approach is to learn directly from expert demonstrations in order to extract implicit task knowledge and strategies. As such, we collect and make available a novel and diverse dataset of bin packing demonstrations by humans in virtual reality. In total, 263 boxes were packed with supermarket-like objects by 43 participants, yielding 4644 object manipulations. This collection is annotated with multiple task parameters and is the most diverse public dataset involving irregular objects. It has the potential to train models that achieve efficient space usage, safe object positioning and to generate human-like behaviors that enhance human-robot trust in a collaborative scenario.

We leverage the data in this new dataset to learn a Markov chain to predict the object packing sequence for a given set of objects. The proposed model makes predictions in real-time, only requires a simple and fast training scheme, and accurately captures the strategies that humans use during the bin packing task. Our experimental results show that the model generates sequence predictions that are indistinguishable from human-generated sequences when classified by individuals.

Keywords

Bin Packing; Learning from Demonstrations; Datasets for Robot Learning; Human-centered Automation

Resumo

Os supermercados exclusivamente online têm expandido consideravelmente nos últimos anos. O seu funcionamento requer embalar e expedir milhões de encomendas da forma mais eficiente possível. A automatização do processo de embalar os produtos tem progredido lentamente, principalmente devido à complexidade inerente a embalar de forma segura objetos irregulares. As restrições que esta tarefa impõe sobre o posicionamento e manipulação dos objetos, e as variadas propriedades físicas dos mesmos tornam ineficazes comportamentos pré-programados.

A abordagem proposta consiste em aprender diretamente de demonstrações, extraíndo conhecimento implícito da tarefa. Como tal, é reunido um novo e diversificado *dataset* de demonstrações de indivíduos a arrumar objetos em realidade virtual. No total, 263 caixas foram arrumadas com objetos de supermercado por 43 participantes, resultando em 4644 manipulações de objetos. Este *dataset* é anotado com múltiplos parâmetros e e atualmente é o mais diversificado entre os que contém objetos irregulares. O dataset tem potencial para treinar modelos para obter um uso eficiente do espaço, gerar posicionamentos seguros para os objetos e comportamentos semelhantes a humanos, aumentando a confiança nesta tarefa de colaboração com robôs.

Utilizando estes dados é treinada uma cadeia de Markov que permite estimar a sequência de arrumação para um conjunto de objetos. O modelo proposto faz estimações em tempo real, apenas requer uma aprendizagem simples e rápida, e captura com exatidão as estratégias que os indivíduos utilizam durante a tarefa. Resultados experimentais revelam que o modelo proposto gera sequências que são indistinguíveis daquelas geradas por humanos.

Palavras Chave

Embalar Produtos; Aprendizagem Através de Demonstrações; *Datasets* para Aprendizagem Robótica; Automatização Centrada em Humanos.

Contents

List of Figures

List of Tables

Acronyms

6-DoF	6 degrees of freedom
ALM	Additive Layer Manufacturing
BoxED	Box packing with Everyday items Dataset
DRL	Deep Reinforcement Learning
IRL	Inverse Reinforcement Learning
LfD	Learning from Demonstrations
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
RNN	Recurrent Neural Network
RL	Reinforcement Learning
seq2seq	Sequence-to-sequence
SPM	Sequential Pattern Mining
Sim2Real	Simulation-to-Reality
VMM	Variable-order Markov Model
VR	Virtual Reality

1

Introduction

Contents

1.1 Motivation

The ceaseless digitalization of the modern world has steadily transformed many aspects of our daily lives, including the grocery shopping experience. In recent years it too has begun the shift from the physical retail spaces into the online environment, with companies such as Ocado [1], a dedicated online grocery retailer, reporting 2.5 billion GBP of revenue in 2021.

However, hidden behind the simplicity and convenience of an online purchase lies a logistics challenge that involves collecting, packing, and shipping the purchased items as efficiently as possible. And even considering that in semi-structured environments such as warehouses the collection of the items can be done (to some extent) by automated mobile robots, the process of packaging multiple, irregularly-shaped items has yet to be automated. This is particularly challenging in supermarket-like environments where there is a massive variety of items ranging from regular cardboard-box cereals to soft, irregular vegetables.

Concurrently, robots have steadily been entering our workplaces and homes. These collaborative robots have attracted a considerable amount of research and development attention, with the introduction of Amazon's new Astro robot [2] and Temi robot [3]. Besides the critical need to ensure a safe operation while interacting with humans, they must also act as human-like as possible, such that their actions can be understood and anticipated by their human owners, providing users with an improved interaction experience [4].

The research project that motivated this thesis lies at the intersection of these two rapidly expanding fields. It is called PackBOT and it is illustrated in Figure 1.1. Its goal is to develop the first framework for a collaborative robot that is able to operate side-by-side with a human worker in order to pack any kind of items in a box, while simultaneously learning from observing the human how to safely pick and place all items.

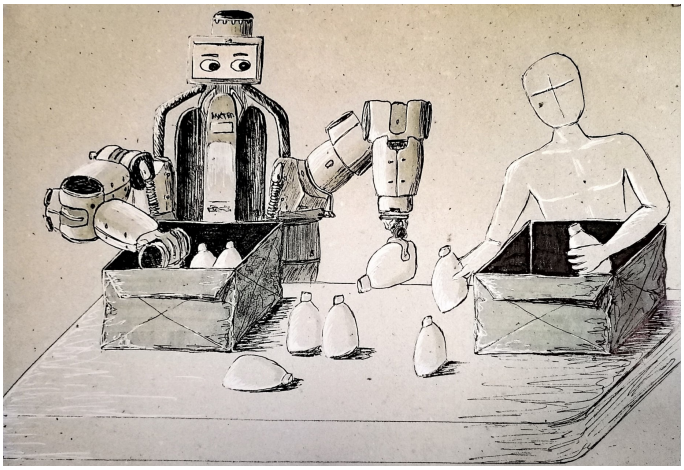


Figure 1.1: Illustration of the PackBOT project.

To achieve these goals, the use of collaborative robots is essential. This necessity is not specific to our goals but rather a pressing need felt both in academic and industrial environments for safer robots that can coexist in the same workspace as other humans. Collaborative robots are an outstanding achievement of modern robotics. Their invention is usually attributed to J. Edward Colgate and Michael Peshkin, who in 1997 filed their patent for *Cobots*, described as

> ”An apparatus and method for direct physical interaction between a person and a general purpose manipulator controlled by a computer.” ([5]).

One conceptualization of such a device can be seen in Figure 1.2. Since then, many technological developments such as better sensors, the introduction of flexible joints, and increased processing power have enabled the development of a vast diversity of collaborative robots. Although they were mainly used in industrial environments, their safety and robustness have improved so much that there has been a growing interest in collaborative robots in the social and domestic robotics areas.

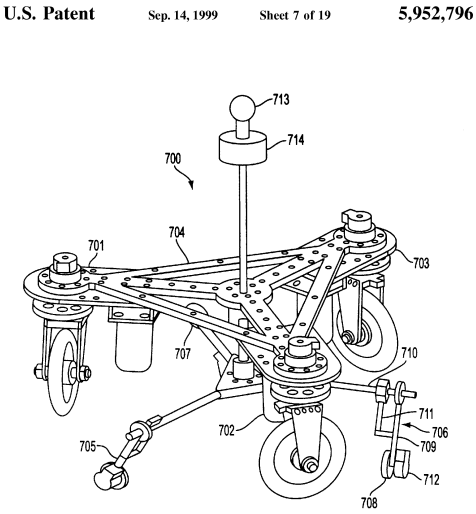


Figure 1.2: Diagram of a three-wheel cobot to illustrate one possible embodiment according to the original patent. The human operator would cooperate with this cobot via the vertical handle. This input would be measured by a computer and sensors to move the cobot accordingly. Adapted from [5].

The introduction of these robots in the highly complex, ever-changing environments that are our homes and workplaces means that it is of the uttermost importance to develop learning mechanisms that make these robots capable of adjusting to changes and capable of autonomous learning. Perhaps even more important is that these cobots are capable of acquiring the user’s trust, which requires that they behave similarly to humans.

1.2 Objective

As mentioned earlier, this thesis is framed within the research project called PackBOT. In this work, we focus on learning aspects of the packing task with irregular objects from human demonstrations. In particular, we seek to predict the packing sequence given the set of objects that needs to be packed. For instance, given an order from a customer at an online supermarket, we want to predict the sequence in which the objects should be packed into the container. This is one of the steps in the bin packing task, which can be roughly decomposed into three main components or sub-tasks.

The first is how to grasp objects with a robotic gripper so that an autonomous agent can pick and place an arbitrary object. Research on software for grasp inference has been extensive and many approaches have achieved impressive accuracy [6]. Furthermore, many different types of grippers have been developed [7], ranging from the more traditional two-finger gripper to soft grippers that have been deployed to grasp soft, irregular, and fragile objects [8]. Given this extensive literature, this thesis does not propose new methodologies for this sub-task. However, the dataset we collect in this work contains data that can be used to learn better grasp inference models with supermarket-like objects.

The second is determining the order in which the objects should be packed into the container. This aspect has been partially overlooked since research into bin packing usually assumes that the objects arrive at the robotic manipulator on a conveyor belt, as is shown in a review of this field in Section 2.3. However, this assumption implies that there is no situation where the robot needs to choose the order in which to pack the objects, or that this order is not important, which is not true in many situations. The research that addresses this issue generally assumes that the objects are cuboids or that they can be approximated as cuboids (as reviewed in Section 2.3), which is only valid for some industrial applications. Given this gap in the current bin packing literature, we propose a strategy that tackles this sub-task.

The third and final sub-task is predicting a placement pose inside the box for each object. This pose must ensure that both the object and all other items already in the box are safe and stable, and that the final layout of the items inside the container uses the available space efficiently. In Section 2.3 a review of the literature on this topic is presented. Although this thesis does not propose a specific method to solve this task, we propose a new dataset that can be used to learn better models for placement pose prediction.

In summary, the contributions of this thesis are threefold:

1. We learn a model to predict the packing sequence, based on Markov chains, that extracts implicit task knowledge directly from human demonstrations. Unlike previous research on this topic, our model is learned and tested with irregular objects and considers their fragilities. Another advantage is that it is much faster at predicting packing sequences and can be deployed in real-time.

2. To learn the aforementioned model we create a new dataset called Box packing with Everyday items Dataset (BoxED), the first publicly available collection of human experts packing groceries into a box. BoxED was collected in Virtual Reality (VR) and captures many parameters of this task, including 6 degrees of freedom (6-DoF) pick-and-place grasp poses, object and headset trajectories, packing sequences, and more. This dataset enables learning models for multiple aspects of this task from humans, which is unlike all other public datasets in the field of bin packing. This dataset is made available in this link[1], along with an object-oriented Python framework that facilitates importing and using these data.

3. The proposed model is validated with real experiments involving human experts. The experimental results reveal that the predicted packing sequences lead to the safe and efficient positioning of the objects and are classified by the experts as indistinguishable from human-generated sequences.

Furthermore, the research developed in this thesis led to:

1. The submission of a short paper and presentation of the associated poster at the 2022 IEEE International Conference on Development and Learning (ICDL), in London.

2. The submission of a paper[2] to the 2023 IEEE International Conference on Robotics and Automation (ICRA), which is currently under review.

1.3 Document Structure

The rest of this document is structured as follows. The following Chapters 2 and 3 lay the theoretical basis required for this thesis. Chapter 4 presents the dataset, namely what it contains and how it was collected. Also in Chapter 4 is the introduction of the proposed model for packing sequence prediction, specifically how it is learned from the expert demonstrations we collected, and how it is tested and validated. Finally, Chapter 5 presents the main conclusions and implications gathered from this work, as well as possible directions for future work.

[1] https://vislab.isr.tecnico.ulisboa.pt/datasets_and_resources/#BoxED
[2] http://arxiv.org/abs/2210.01645

2

Background

Contents

One may argue that collaborative robots have been closely coupled with the development of robots. In fact, some of the first robots ever made arose from a need to handle radioactive materials from a safe distance. These pioneer robots built in the early 1950s hardly fit in our current concept of a robot - they were master-slave manipulators that had simple controls and almost no autonomy (Figure 2.1). Nevertheless, they evolved from the need for machinery that would *cooperate* with a human to accomplish a task that would otherwise be impossible.

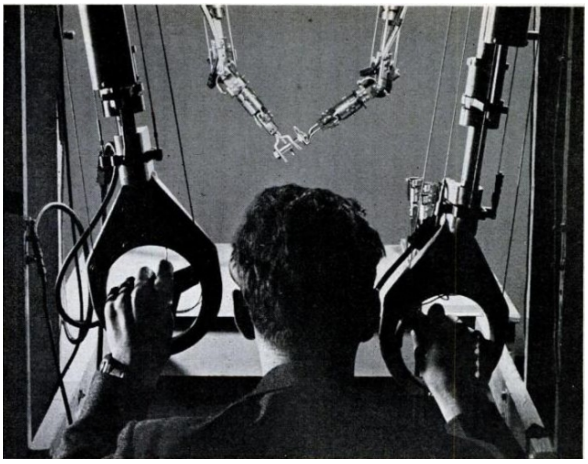


Figure 2.1: Example of master-slave manipulator developed in the 1950s. Adapted from [9].

Although these early "robots" already revealed in some ways the need for cooperation between humans and robots, they were devoid of any intelligent capabilities. In the following decades, many technological developments enabled the construction of much more complex robots, capable of performing more diverse tasks compared to the simple master-slave manipulators. Furthermore, the increase in computational power and data availability enabled the development of more flexible learning frameworks, adequate for a larger variety of tasks.

The contents of this chapter are active research fields that branched from these pioneer robots. The chapter presents an introduction to relevant approaches in the field of Learning from Demonstrations (LfD) and high-level task learning, two important aspects of any collaborative robot. After an overview of LfD methods, a review of sequence prediction strategies is presented. Then follows a summary of approaches to packing objects into a container, followed by a review of available datasets which are relevant to the goals of the thesis.

2.1 Learning from Demonstrations

Learning from demonstrations consists of learning how to perform new tasks and their constraints by observing an expert complete the task first. This paradigm has attracted increasing attention from

the robotics community because it circumvents predefined behaviors and introduces more flexibility in robotic learning. In this thesis, we seek to learn certain aspects of the packing task and its underlying constraints from passive observations of an expert.

One common approach for extracting knowledge from demonstrations is policy learning, which can encompass algorithms from Reinforcement Learning (RL), supervised learning [10] and Inverse Reinforcement Learning (IRL) [11]. A policy $\pi : S \rightarrow A$ maps the input space S (usually the state space) into the output space A (usually the action space). The goal is to learn a policy that can accurately mimic the demonstrations or to obtain a policy that learns underlying rules from the demonstrations and is able to replicate the desired outcome.

The formulation above assumes that there is an external signal that indicates which states are desirable and which are not, thus guiding the policy learning process. This is the reward signal, usually denoted as $R_a(s, s')$, and provides the reward for transitioning from state s to state s' due to action a. This is essentially the formulation of a Markov Decision Process (MDP), which is an underlying building block for much of the research in this field [12–14].

An MDP models decision-making in situations where there is uncertainty in the actions' effects and where different states have different rewards or desirabilities. Generally, the policy is learned by maximizing the total reward that it achieves. Its well-established mathematical foundations make it a popular choice for modeling human-robot interactions. For instance, Munzer T. et al. [12] used a variation of Markov Decision Processes along with first-order logic to develop a system for a collaborative robot that learns tasks and human preferences before and during execution.

One major difficulty when learning a policy from demonstrations under this formulation is how to encode the demonstration in a format that can be processed by the learning algorithm. In the case of passive observation, the information can be a video of the task execution which presents additional difficulties for representation extraction. To address this issue some works introduce simplifying measures, such as having the expert wear some form of motion capture suit [15] or having the demonstrations in a simulated environment [13], while others use complex pose estimators [16]. Once the relevant information has been extracted from the demonstrations, it may still be necessary to obtain a compact representation of the data. For instance, in a work by Dasari S. and Gupta A. [13], a transformer network [17] is deployed to extract a state representation from a video of the demonstration.

Instead of introducing simplifying assumptions or extracting the state of the system from the demonstrations, some approaches bypass the explicit characterization of the state by defining as input the raw data from the system [14, 18, 19]. These approaches use neural networks to learn end-to-end policies that extract meaningful information from the raw input data and map it to output actions. This type of implicit policy learning is useful to circumvent state characterization in tasks where defining the state is challenging. However, end-to-end policies require large amounts of data, which may be expensive for

specific tasks, and prevent most methods of theoretical analysis for establishing performance guarantees.

In the context of learning the bin packing task from human demonstrations, collecting a dataset that is large enough to train an end-to-end policy with neural networks would be impractical and very time-consuming, which is unfeasible in the time frame allocated for this thesis. Furthermore, a key parameter in this task is the placement pose of each object inside the container. There are two approaches to accurately recording the poses of all objects during the task. The first would be to use a motion capture system along with reflective markers on each object. However, this would still be susceptible to occlusions and the markers could potentially influence how individuals handle the objects. Furthermore, it would also be necessary to track the pose of the hand of the participant to record the grasp poses.

The second approach is to conduct and collect the demonstrations in a virtual environment using VR technology. This is the approach that is chosen in this thesis since it provides the exact values of all the parameters and variables we need to collect. Furthermore, we can also design a virtual gripper that resembles a robot's gripper and thus minimize the domain gap between the demonstrator and the robot.

2.2 Sequence Prediction

Within the field of sequence prediction there are multiple types of sub-tasks, for instance:

- Sequence-to-sequence (seq2seq) problems where the goal, as the name suggests, is to predict an output sequence of tokens given another input sequence. In the most general case, both the input and output sequences can have any length. Some examples of this type of problem are machine translation tasks and conversational agents.
- Sequence classification problems that aim at labeling an entire sequence as belonging to a single class. For instance, in sentiment analysis the goal is to classify a review as good, bad, or according to other descriptors.

In the context of box packing, the sub-task that is most related to sequence prediction is predicting the packing sequence. Its goal is to predict which object should be placed next inside the box, given what objects are still available to pack and what objects have already been packed. For this task, the most relevant type of sequence prediction is next value prediction. This type of sequence prediction focuses on predicting the next element of a sequence of tokens and encompasses problems such as time series forecasting and product recommendation.

Algorithms that address this class of problems range widely in their approach, from explicit association rules and Sequential Pattern Mining (SPM) algorithms [20, 21], to deep learning based approaches which learn implicit representations [22, 23]. These algorithms search for sub-sequences that appear

often in the data and thus contain important sequential or associative information to predict the next element. The former methods, sequential pattern mining approaches, benefit from learning models that are human-interpretable and are described in Section 3.2. This section will focus instead on introducing deep learning approaches.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [24] were purposely designed to model sequential data. These are the standard deep learning choices for the task of next value prediction. For instance, in [23] the authors train both a RNN and a LSTM network to predict the price of stocks at every minute.

More recently, Transformer networks [17] have gained popularity for their performance in sequence modeling tasks, including next value prediction. This advantage is partially due to its ability to retain information from more distant parts of a sequence and process longer sequences than RNNs or LSTMs, which is a consequence of its attention layers. More concretely, these networks only require one operation to relate two words, regardless of how far apart in the sequence they are. Recent models, such as BERT [22], are trained for language modeling on very large datasets with outstanding performance. These models are capable of generating long segments of text through next word prediction.

However, neural networks for sequence prediction such as these require large datasets. For instance, the stock price predicting RNN [23] was trained on more than three months of stock prices sampled every minute. Even larger is the training dataset of BERT [22], which contained billions of words. Although these models are capable of achieving record-breaking performances, such large datasets are not available for most tasks such as box packing. For this reason, this thesis deploys alternative methods (described in Section 3.2) to learn and predict the packing sequences from expert demonstrations.

2.3 Bin Packing

The field of bin packing addresses the task of packing a set of objects into one or more bins with maximum space efficiency and object safety. Although this field hasn't been substantially researched, the need for such methods is increasingly pressing. Grocery shopping is transitioning to the online space which puts growing pressure on supermarkets to pack and ship online orders.

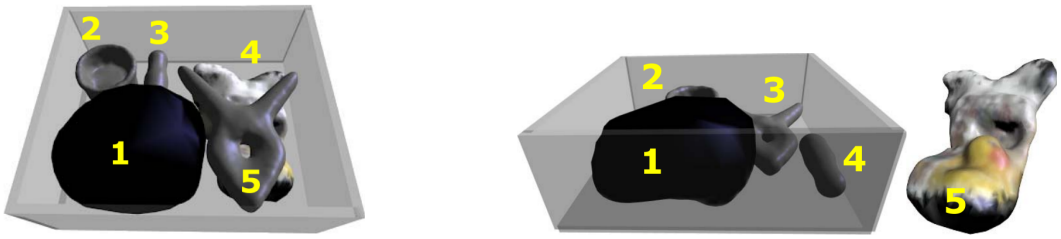
There are two main types of tasks in this field since the packing process can be either offline or online [25]. In the offline bin packing problem the goal is to pack a set of objects into a container and all the objects are available from the start. This includes two sub-tasks: choosing a packing order and predicting a placement for each object. On the other hand, the online bin packing problem assumes that the objects arrive one by one (for instance, on a conveyor belt) and need to be immediately packed into the container. Thus, it mostly corresponds to the second stage of the offline problem.

The following review will first address the approaches that include the sub-task of choosing a packing

sequence and afterward those that focus solely on predicting a placement pose. This distinction is made because considering sequential data drastically changes the type of method that can be deployed. It concludes by reviewing the first research on a prototype for an automatic supermarket checkout system that packs groceries.

2.3.1 Packing Sequence Prediction

Although it appears deceivingly simple, the sequence in which a set of objects is packed into a container can negatively impact how good the final layout of the objects is. This is illustrated in Figure 2.2, where two sequences for the same set of objects lead to different outcomes. In one situation, due to a sub-optimal sequence, the packing planner was unable to find a solution.



(a) (b)

Figure 2.2: The predicted packing sequence can negatively impact the success of the packing task. In (a) the planner successfully packed all objects in the container whereas in (b), for the same set of objects, a different packing sequence leads to failure as there is insufficient space inside the box for the last object. Both images were adapted from [26].

As such it is important to predict a packing sequence that enables a safe, stable, and efficient placement of the objects. Most recent approaches to this sub-task do not focus solely on predicting the packing sequence. Rather, they deploy a Deep Reinforcement Learning (DRL) agent to jointly predict the packing sequence along with another parameter, such as the orientation of the placement [27, 28], or the position and orientation of the placement simultaneously [29]. These approaches formulate the problem as a sequence-to-sequence prediction task where the input sequence contains all the objects, and optionally some of their properties (e.g., dimensions), and the output sequence corresponds to the predicted best packing sequence.

The neural networks trained in these approaches follow an encoder-decoder shape and are trained on a RL scheme to learn a policy that predicts the probability of the next object in the sequence in each decoding step. An example of this structure which was proposed by [29] is presented in Figure 2.3.

The network in Figure 2.3 consists of two Recurrent Neural Networks, one for the encoder and the other for the decoder. The encoder network receives, at each step, the dimensions of one of the objects that should be packed as input. The decoder receives the output from the final encoding step

of the encoder and the prediction made by the decoder in the previous decoding step. Furthermore, the authors also deploy an attention mechanism to better integrate the information from the encoder and decoder when predicting the next item in the sequence. The inclusion of an attention mechanism is a common feature in these architectures [27, 28].

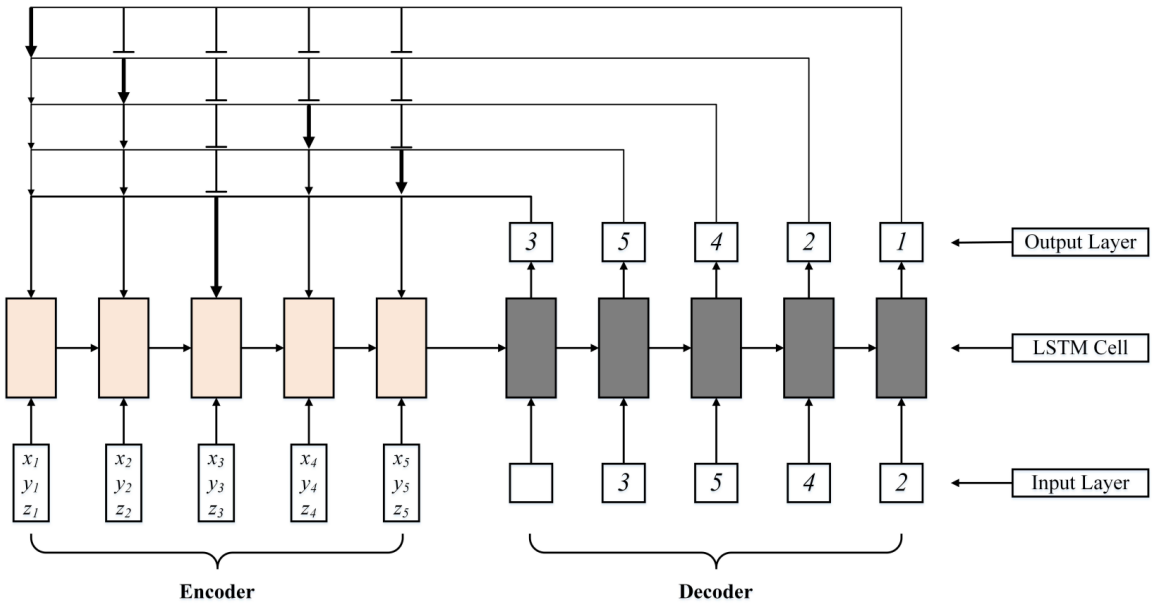


Figure 2.3: Neural network illustrating the common encoder-decoder structure proposed by [29] to predict the packing sequence. Image adapted from [29].

In [27] the authors predict the orientation of the placement along with the packing sequence, also using an encoder-decoder neural network. To decouple the placement orientation from the placement position, the proposed approach uses a search strategy with a custom heuristic to predict the placement position.

Even though this type of approach for sequence prediction can lead to good performance, they generally assume that the objects are cuboids. Although this assumption is valid in some industrial applications, in most situations with everyday objects it is not. Consequently, many of these methods cannot be applied to everyday problems such as the grocery packing problem. Furthermore, the reinforcement learning scheme is not guaranteed nor encouraged to produce human-like behaviors, which is inadequate for applications that involve human-robot interaction.

Since the approach proposed in this thesis involves collecting data from human demonstrations, we must consider a fixed-size collection of objects. Although this assumption restricts the diversity of objects that the proposed model considers, it is a trade-off that enables learning a model for diverse object geometries which also considers their physical properties (such as fragility and weight), unlike previous research in this field.

2.3.2 Placement Pose Prediction

So far in this review, the bin packing task has been described under the context of packing objects one by one. However, a large portion of research [30–32] actually focuses on a slightly different problem: packing all the objects at once. This seems counterintuitive because the goal is not to pack the objects for shipping, for instance, but rather to optimize their layout such that the total volume of the objects is minimized. This is an important problem in some industrial applications such as Additive Layer Manufacturing (ALM), which is the industrial name for 3D printing.

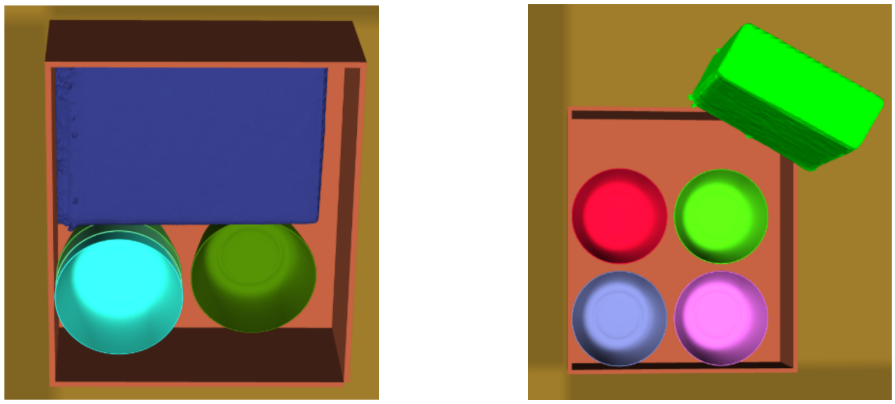
In this context, it is beneficial to minimize the total volume since this also minimizes the amount of material that is wasted. Furthermore, time constraints are not imperative since the manufacturing processes can be planned ahead of time. However, deploying these methods in the task of packing groceries would lead to a different problem: reverse-engineering a packing sequence that would lead to that final layout of the objects. Since this is an equally complex problem and not guaranteed to have a solution, these approaches are not adequate for the task addressed in this thesis. As such, the following review focuses on placement prediction methods that can be deployed in the grocery packing task.

In general, there are two categories of approaches, within the aforementioned context, to predict the placement pose for an object in the packing task: those based on search strategies and those that use learning methods with limited supervision such as reinforcement learning. The first cluster has the advantage of fast deployment as there is no learning phase, however, most have slow execution times (ranging from dozens of seconds to a few hours [26, 33]). Conversely, methods based on RL or DRL [34,35] require complex training schemes but have faster execution times. In this section we dissect some of these approaches along with their advantages and disadvantages.

Search-based strategies seek to formulate the packing task as an optimization problem under task-specific constraints and guided by heuristics. The constraints can restrict the placement positions for an object (for instance, all objects must be placed in the internal space of the container), enforce stability requirements for the final pose of the object, or impose manipulation feasibility. As an example, in [26] the authors formulate three constraints, two of which focus on the valid positioning of each object such that it does not collide with other objects but is stable under gravitational and frictional forces. The third constraint focuses on ensuring that a planned manipulation is feasible and does not disturb other items in the container considering the robotic manipulator. Just verifying these constraints requires solving a convex optimization problem and performing collision checking for multiple sampled poses during the manipulation of each object.

On the contrary, heuristics do not seek to restrict the search space, rather they intend to guide the search to optimal solutions. They are usually formulated such that the search algorithm gives preference to placement poses that lead to more efficient space usage. As such, heuristics play a key role in the

success of a search-based algorithm. For instance, F. Wang and K. Hauser [33] formulate a heuristic that, given the heightmap of the container, guides the search towards solutions that minimize the occupied volume and maximize overlap with the surface below the object. As such, the proposed approach leads to more efficient space usage than previous approaches. As shown in Figure 2.4, the proposed heuristic leads to a solution that stacks the bowls in order to minimize the occupied volume, whereas a previous heuristic leads to poor space usage.



(a) **(b)**

Figure 2.4: Final container layout after solving the packing task with two different heuristics. (a) Heuristic proposed by [33]. (b) Another heuristic that does not optimize for space efficiency. Both images were adapted from [33].

The advantage of these approaches is that they can be constructed as object-agnostic approaches (i.e., do not require explicit knowledge of the object models). However, both the constraints and heuristics must be hand-crafted in a time and labor-intensive process. Furthermore, since the search space for each placement pose is six-dimensional, the search process can be very time-consuming.

Strategies based on reinforcement learning formulate the packing problem as a Markov Decision Process. In this formulation, the state usually encodes information about the objects inside the box and the characteristics of the object to pack next, and the chosen action is the predicted position for the next object. Since a position inside the container is a continuous 3-dimensional variable, many methods discretize the volume into a grid where each element is a possible placement position [36]. This discretization is a balance of accuracy and complexity since better accuracy requires a smaller grid size which drastically increases the number of grid elements.

To avoid this problem, in [34] the authors propose a new tree-based representation of the box and the objects in it. The proposed approach is the first to consider a continuous solution space and achieves superior performance. However, the approach considers only cuboid objects and requires the inclusion of other components (namely Graph Attention Networks) to extract compact representations from the tree representation.

Once the problem has been formulated, DRL approaches will usually train a deep learning agent to map the input state space into a target output space. This later generally encodes the action space, and as such, the agent predicts a placement pose for the next object [34]. However, in [35] the input to this deep neural network, represented in Figure 2.5, is a set of candidate placement poses for the next object represented as potential future states of the box, and the output is an estimate of the quality of each candidate. An advantage of this approach is a considerable reduction of the dimensions of the input and output spaces, thus making the training process less complex.

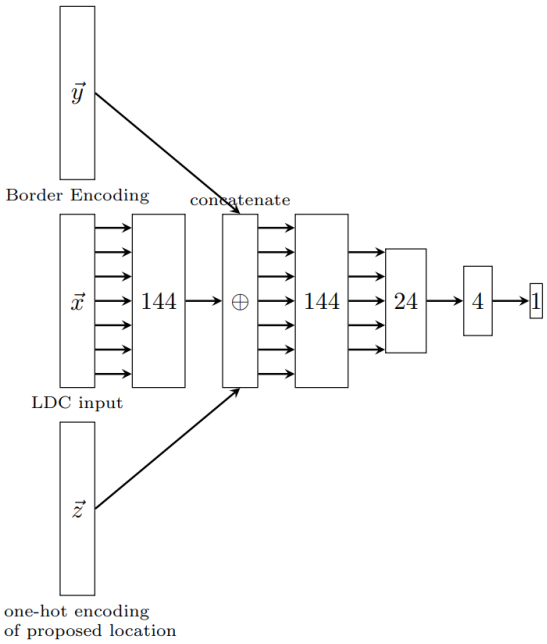


Figure 2.5: Neural network deployed in [34] to estimate the quality of possible placement poses for an object. Image adapted from [34].

Unlike the approaches described in this section, in this thesis we do not propose a solution for predicting the placement pose of each object. However, we argue that the data we collect has the potential to create a model with superior performance that can handle irregular objects and lead to a safe positioning of all objects, regardless of their fragilities. As these data are collected on multiple objects with different geometries and properties, they can be used to learn models for packing not just cuboid objects but also irregular, non-convex objects. Furthermore, learning directly from humans can circumvent the convergence difficulties exhibited when training a DRL agent on a large action space [36].

2.3.3 Complete System for Bin Packing Groceries

Recently, Y. Aquilina and M. A. Salib [37] proposed the first prototype for an automated checkout system. The system, shown in Figure 2.6, consists of a conveyor belt that transports the client's products towards

the packing robot, which is equipped with a vacuum gripper. The products are picked by the gripper and placed in a box.

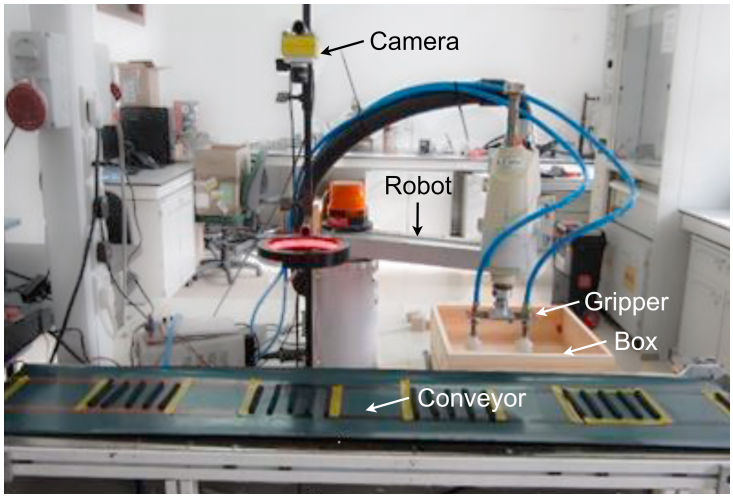


Figure 2.6: First prototype for an automated supermarket checkout robot proposed by [37]. Image adapted from [37].

In the experiments reported by the authors, the complete system took about 20 seconds to pack each item, on average. For comparison, in our virtual reality experiments, the participants took 4 seconds to pack each object, on average. The authors successfully showed that the concept of an integrated system for this task was feasible. However, there are important limitations to this system. For instance, although it handles somewhat diverse objects, it requires that the objects are flat enough to be grasped by the vacuum gripper. Furthermore, the system relies on prior information about the objects and also requires that the objects are labeled with a large bar code visible from above. Finally, the packing algorithm only considers object geometry and ignores other important characteristics such as fragility. Nevertheless, the system was a successful proof-of-concept while highlighting that further developments are needed in the field of bin packing to achieve an autonomous system that can be deployed in real-life situations.

2.4 Related Datasets

This section reviews public datasets that contain relevant data to the task addressed in this thesis. Unfortunately, there are very few datasets in that category. Perhaps most relevant is the dataset proposed by Song S. et al. [38], which addresses grasping irregular objects in a scene that mimics an everyday environment, such as a kitchen or an office. It consists of videos of eight participants performing pick-and-place tasks in cluttered environments using the custom-made gripper shown in Figure 2.7, which is similar to a robotic gripper.

The purpose of this custom-made gripper is to minimize the domain gap between the human demon-

strations and the hardware that the robot will use when learning from these demonstrations. The videos are annotated with the 6-DoF gripper trajectory (before the grasp and during the object manipulation), 6-DoF grasp pose, picking order, and object mask.

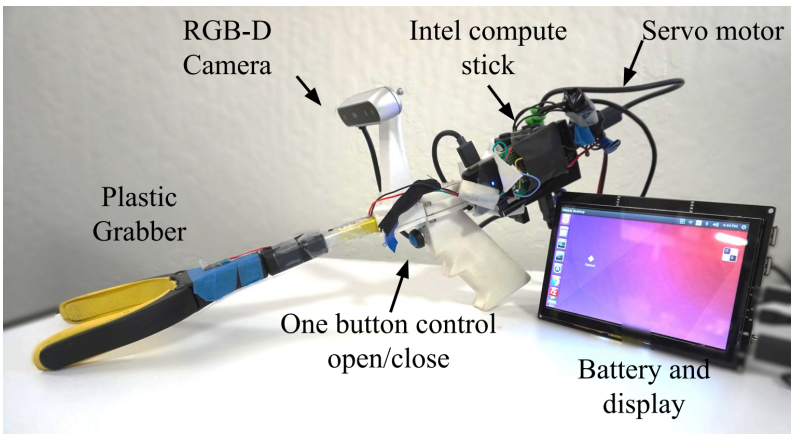


Figure 2.7: Custom-made gripper developed for data collection by [38]. The gripper is equipped with an RGB-D camera and a servo motor for automatic finger closing. Image adapted from [38].

Although more encompassing than most robotic grasping datasets, this dataset is not related to bin packing and as such does not include packing order nor placement inside a container. Additionally, since these data are collected in a real environment, the videos are also not annotated with the objects' pose because pose estimation in a real environment is not trivial due to occlusions and fast movements.

Other research on the bin packing problem usually deploys reinforcement learning or heuristic-based strategies and does not require annotated datasets [33–35]. Those that deploy supervised learning use datasets that consist of industrial applications which generally consider cuboid objects and industrial environments [27]. Consequently, these datasets cannot be deployed for our bin packing task.

This encouraged us to collect and make available a new dataset that collects data from humans packing grocery objects into a container, which has not yet been done. Due to the aforementioned difficulties in estimating an object's pose in a real environment, this dataset is collected in VR. These data are more diverse and encompassing than previous datasets, both in terms of participant diversity and in the number of parameters that are recorded.

3

Methodologies

Contents

This chapter provides a detailed description of the methods that play a key role in achieving the thesis's goals. Here we address tools such as Unity, used to create the virtual environment where all experiments take place, and methods including Markov chains and sequential pattern mining algorithms used to predict the object packing sequence from the data we collect. The chapter concludes by introducing relevant statistical tests used to measure the significance of our experimental results.

3.1 Virtual Reality as a Tool for Data Collection

The peculiar task of supermarket-like object packing has not yet been extensively explored in research. As such, it is necessary to first collect data from expert demonstrations. Collecting these data will inevitably require object classification and segmentation, grasp pose detection (which requires hand tracking), pick and place pose detection, and finally object tracking to save the trajectory of each object. Collecting all these variables in a real environment would require multiple complex systems, similar to [38]. These issues can be circumvented by collecting the data in Virtual Reality (VR) since the exact value for each variable is always known. However, VR is not devoid of its limitations. Section 4.1.3 provides a thorough analysis of the limitations introduced by this technology and explains why they are neglectable.

The real-time development platform Unity [39] plays a key role in achieving this, as it is the chosen platform to develop all of the virtual reality components and experiments of this thesis. It enables the development of a 3D virtual world that can be compiled and tested in real-time and provides the flexibility to implement custom functionalities. This section provides a brief overview of some of Unity's core functionalities, which are useful for this thesis and possibly for other projects involving human-object interactions. Note that the following analysis is merely an introductory overview since an in-depth explanation is outside the scope of this document.

3.1.1 Bypassing Object Trackers and Pose Estimators

As previously mentioned, many of the disadvantages and challenges associated with tracking human-robot and human-object interactions in real life can be overcome by analyzing them in a simulated environment. This is mainly because the ground truth pose of all items in the virtual environment is known at every instance.

If the environment is developed in Unity, then each object will have its pose available for access at any instance. More specifically, every pose is specified with respect to the origin reference frame automatically defined by Unity at the center of the environment. A peculiar aspect of Unity's coordinate frames is that the Y-axis points up. For each object the developer can access every property needed to

define its state in the environment. These include position, orientation, linear and angular velocity, and more.

Some of these properties are easily accessible and modifiable through the graphical editor as shown in Figure 3.1. However, the straightforward access to all properties that Unity provides through scripting is perhaps more useful. All objects in the virtual environment can be attached with custom scripts that will be able to access their properties and modify how the objects behave in the environment. Through this functionality it is possible to access and record an object's pose, for instance, at any desired sampling rate. Note though, that this rate is always constrained to the refresh rate of Unity's physics engine, which can be customized within the capabilities of the machine on which the environment is being deployed.

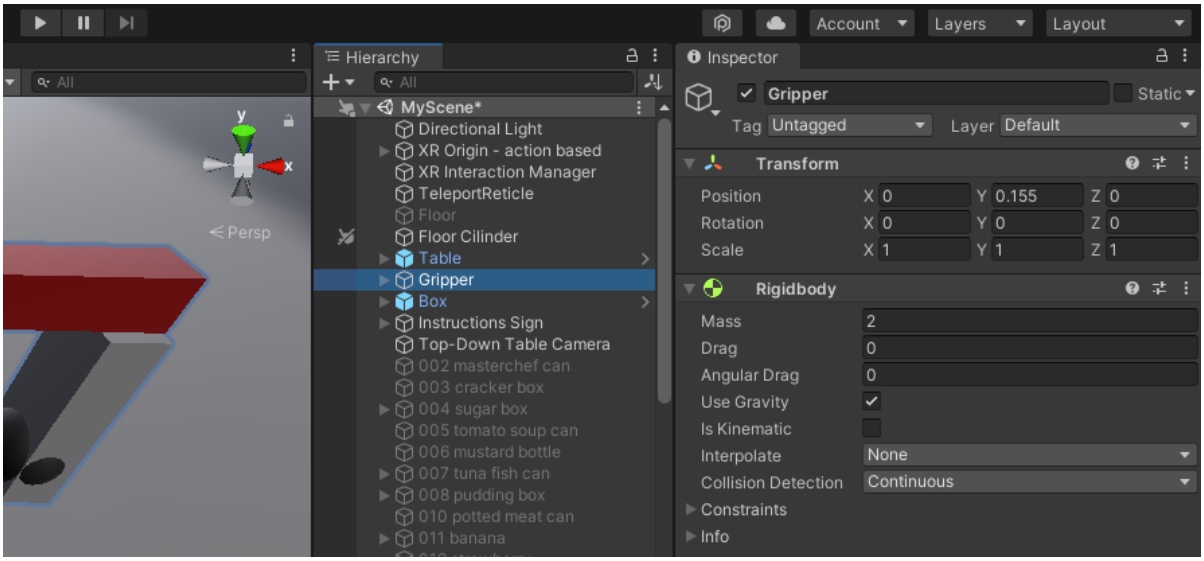


Figure 3.1: A screenshot of the Unity Editor displaying, on the right side of the figure, the current position, orientation, and more properties of an object (the virtual gripper, in this case).

3.1.2 Physics Engine

Perhaps one of the most fundamental components is Unity's physics engine, which drastically simplifies the setup of a simple virtual environment that feels mostly real to the user. Unity uses NVIDIA PhysX [40] as its 3D physics engine and provides multiple customization features. This integration automatically detects collisions and simulates gravity, friction, and other physical aspects of the virtual scene with minimal user input.

One key aspect is parenting objects to other objects. This enables the creation of complex objects that are composed of multiple parts but behave as a whole. This can be especially useful to create objects with moving parts, such as the virtual gripper.

To obtain realistic object behavior, Unity provides the option to configure objects as rigid bodies. This

assigns the object multiple physical properties, namely a mass, drag coefficients, and more. Adequate configuration of each rigid body's properties is important to achieve realistic object interactions, a key aspect when performing experiments that analyze how humans manipulate different objects.

However, configuring an object as a rigid body does not provide it with the capability of colliding with other objects. To enable collisions, each object must have its own collider. In Unity, a collider specifies to the physics engine what the object's boundaries are, such that the engine knows where to search for collisions. Thus, collisions are detected between two colliders and not between two object meshes.

Unity provides only a few basic shape colliders which are the box, sphere, and capsule colliders. It also provides a collider for complex shapes, which is the mesh collider. However this will approximate the object mesh with a convex hull composed of, at most, 255 triangles, and this collider will also lead to a significant increase in computation overhead. Another option is to approximate an object with multiple basic colliders, as illustrated in Figure 3.2. This often leads to faster performance at the cost of accuracy in collision detection. In this work, both solutions are deployed since objects whose shapes are roughly convex can be approximated with a mesh collider, but non-convex objects require multiple basic colliders.

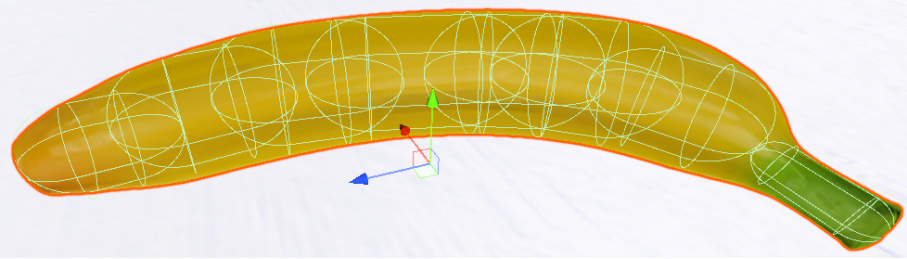


Figure 3.2: An example of a simplified compound object collider. In this example, the shape of the banana is approximated with multiple capsule colliders, which are shown in green lines.

3.2 Sequential Pattern Mining

Data mining is a multidisciplinary field that seeks to uncover patterns, irregularities, or other forms of knowledge from large datasets by deploying different methodologies mainly from machine learning and statistics. Contained within this broad field is the subfield of sequential pattern mining. Methods in this research area address the task of extracting meaningful patterns or rules from ordered sequential data such as time series of stock trading activity or DNA and protein sequences. An important advantage of these methods over more recent deep learning approaches is that they generate models and rules that are easy to interpret. On the other hand, deep learning approaches benefit from large datasets and generally achieve better performance [41].

This section considers as a running example basket analysis - the task of extracting sets of objects frequently bought together from a dataset of multiple baskets. This problem actually belongs to the field

of frequent itemset mining, which is essentially identical to SPM except that there is no order in the lists of items. The reason for using it as an example is that a foundational algorithm for basket analysis, the Apriori algorithm [20], introduced an important property that became an underlying motivation for many of the algorithms in SPM [42]. This property is also an important piece of how we extract knowledge from our dataset in this thesis.

Consider a supermarket owner who wants to optimize the positioning of the products in the store to maximize purchases. A reasonable choice is to place items that are frequently bought together physically close to each other. But how would the supermarket owner extract the sets of objects that are most frequently bought together from the collection of client baskets? In some aspects, the goal is similar to that of a principal component analysis: to extract the underlying components from these data that explain the majority of customers' shopping choices.

Consider as an example the collection of product baskets presented in Table 3.1. The brute-force approach would be to search and store all possible combinations of two products in a basket, then of three products, and so forth, and simply keep count of which combinations appear most often. This is not just inelegant but also practically unfeasible since a simple basket containing 10 items would lead to more than 1000 possible product combinations.

Table 3.1: Example of basket data used for the task of basket analysis.

	Product basket
Client 1	Apple, banana, crackers, tuna can
Client 2	Rice, crackers, wine, cheese, potato
Client 3	Wine, onions, cheese
Client 4	Crackers, banana, wine

A simple and effective observation is that if a set of two products does not appear often in the basket collection then any set of three products that contains this set of two products will also be rare, and thus, irrelevant to the goal. Likewise, if a set of three objects does not occur often in the baskets, then all sets of four items than contain this rare set of three items will also be unlikely. And so, instead of counting all possible combinations of items, one could start by counting how often each item appears in the data and determining which items are frequent. The next step would be to count all possible combinations of two items while discarding those that include one of the rare items, which would drastically speed up the search and reduce the memory complexity. This would be repeated until all combinations were exhausted.

This observation is essentially the intuition behind the well-established Apriori algorithm [20] and many algorithms in SPM [42]. The Apriori algorithm defines as support the probability of an item or set of items appearing in the baskets. One of its key parameters is the minimum support, which is an

adjustable threshold that defines what was informally referred to as ”not appearing often in the baskets”. It is the minimum probability of occurring in each basket for a set of items to be considered relevant.

In the first iteration, the algorithm will traverse the database and keep track of all items and their occurrence probabilities. Then the items with a support probability that is less than the minimum support are discarded and the algorithm repeats the first step for sets of two items (ignoring the discarded items). The pairs of items with a support probability that is less than the minimum support are discarded and the process repeats until all possible sets of items have been discovered.

Consider the example basket data in Table 3.1 and suppose that we define as minimum support a probability of 0.5, meaning that a set of products is relevant if it appears in, at least, half the baskets in the dataset. During the first iteration the algorithm would produce the item count shown in Table 3.2.

Table 3.2: Output of the first iteration of the Apriori algorithm.

Item	Apple	Banana	Crackers	Tuna can	Rice	Wine	Cheese	Potato	Onions
Count	1	2	3	1	1	3	2	1	1
Support probability	0.25	0.5	0.75	0.25	0.25	0.75	0.5	0.25	0.25

In the second iteration of the algorithm, only the banana, crackers, wine, and cheese are considered. The search would look for sets of two of these items that appear often in Table 3.1, such as the set composed of wine and cheese. This process repeats until all possible sets of items have been extracted.

Once the frequent set mining stage of the algorithm is complete, the Apriori algorithm produces patterns or association rules with the format $Item_A \Rightarrow Item_B$, which indicate that when $Item_A$ appears, then it is likely that $Item_B$ will also occur. These rules are labeled with two metrics, confidence and lift. The confidence is how likely $Item_B$ is given that the customer chose $Item_A$. The lift of a rule is how correlated the items in the rule are, and indicates how buying one of the items influences the possibility of choosing the other.

Although the Apriori algorithm cannot be directly applied to the packing sequence prediction problem since it does not consider the order of a list, the important property that it highlights and the concept of minimum support are the basis of the approach that is proposed in this thesis. Since its introduction in 1994, other methods have been proposed that were based on it but introduced extensions to consider the order of a list. Some of these methods, such as the SPADE algorithm [21], also improved the efficiency of the search process, mainly by reducing the number of database scans that are required.

3.3 Markov Chains

A Markov chain is a well-established mathematical framework that models stochastic sequential processes. More specifically, it models processes where each change, or event, depends solely on the current state of the system. This characteristic is known as the Markov property and essentially means that the system has no memory of past events, and consequently, past events cannot influence future changes to the system, only the present state can. Note that this discussion does not consider Variable-order Markov Models (VMMs), which are an extension of Markov chains.

Let s_i define the state of the system at step i. The aforementioned property corresponds to

$$p(s_{i+1}|s_i, s_{i-1}, ..., s_1) = p(s_{i+1}|s_i). \tag{3.1}$$

As such, a Markov chain is simply described by the set of possible states and the probabilities of transitioning from one state to another, as illustrated in Figure 3.3.

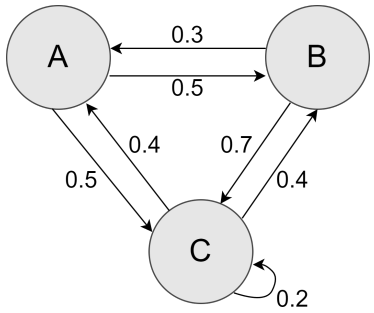


Figure 3.3: An example of a simple Markov chain with three states (A, B, and C), and corresponding transition probabilities.

Its simplicity and effectiveness have made it a popular framework in modeling many systems where the Markov property holds [43]. One notorious example is PageRank [44], developed in 1998 by Sergey Brin and Larry Page which would later become the foundation of Google. This web-page ranking system essentially considers each website as a node of a Markov chain and the links to these websites are the transitions between nodes. This large Markov chain is then processed to infer which website is most relevant.

However, the Markov property is not always applicable since only simple systems display memoryless behavior. Yet many systems can be modeled under the simplifying assumption that they respect this property when in reality they do not. One such example, which displays some resemblance to the box packing task, is next word prediction where the goal is to predict the next word given a portion of a sentence. The next word clearly depends on more than just the previous word, however by assuming that as a simplification, the Markov property holds, a Markov chain can be learned to complete this task.

It should be noted though, that this approach is an oversimplification of the task and would likely not produce the most coherent sentences. Even so, it is useful to illustrate that some complex problems can benefit from the simplification implied by the Markov property. For instance, in the context of this thesis' goals, some aspects of the box packing task can be efficiently modeled under this assumption. One example is predicting the packing sequence for a set of objects, which we address in Section 4.3.

3.4 Beam Search

Consider the following situation. There is a model trained to predict the optimal packing sequence for a set of objects. In this simplistic example, consider also that there are only three different objects, obj_A, obj_B, and obj_C. At each step the model predicts the likelihood of the next object in the packing sequence as a probability distribution over the set of the three objects.

For instance, consider that there are four objects on a table that need to be packed, obj_A, obj_A, obj_B, and obj_C (note that there are two instances of obj_A). In the first step, the model predicts the probability distribution in Table 3.3 for the first object in the sequence.

Table 3.3: Example probability distribution for the first element in the sequence.

Object	obj_A	obj_B	obj_C
Probability	0.2	0.5	0.3

At this point, the sequence prediction algorithm would have to decide which object should be chosen as the first element in the sequence. The obvious answer is to choose obj_B since it is the most likely. This strategy is called greedy search since at each step the algorithm simply chooses the best element and produces good results for very short sequences. However, as the sequence grows, choosing the best element at each step will lead to suboptimal predictions. This is because each element is not isolated in the sequence. Instead, the likelihood of a small sequence segment around an element depends on the elements before it and after it.

This is the motivation for beam search. This search strategy considers multiple possible sequences at each step. The only adjustable parameter is the beam width, which corresponds to how many sequence hypotheses are maintained during the search. Its effectiveness is more easily demonstrated with an example. Consider the situation illustrated in Figure 3.4, where we consider greedy search and beam search with a beam width of 2. Each table represents the probability distribution over the set of objects for the next element in the sequence.

In this situation, greedy search would choose as first element obj_B with a likelihood of 0.5, and then choose obj_C as the second element in the sequence with a likelihood of 0.4. This sequence would then have a total likelihood of $0.5 \times 0.4 = 0.2$.

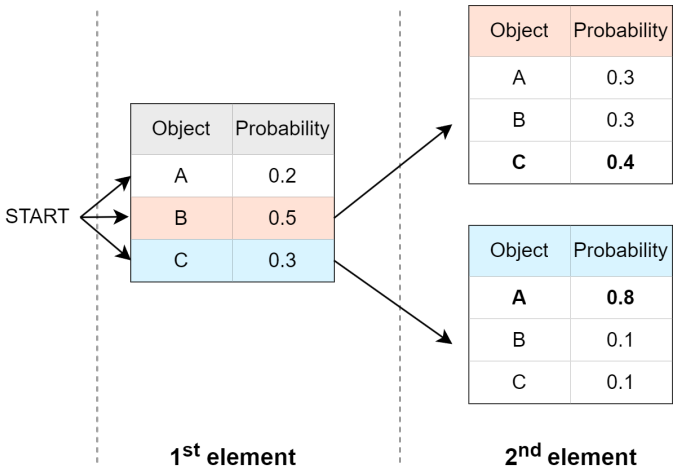


Figure 3.4: Sequence prediction example.

On the other hand, beam search would maintain two possible sequences throughout the search process. In the first step it would choose obj_B as the first element of the first sequence and obj_C as the first element of the second sequence. Then, in the second step these would be extended with obj_C and obj_A respectively. At this point, the most likely sequence would be chosen as the final prediction, which is the second sequence composed by $obj_C \rightarrow obj_A$, with a total likelihood of $0.3 \times 0.8 = 0.24$, which is higher than the likelihood of the sequence returned by greedy search.

The superior performance of beam search over greedy search has made it the default choice for the majority of tasks that involve natural language modeling. This dominance is also due to its customizable trade-off between accuracy and performance. A higher value for the beam width parameter increases computational cost and accuracy, whereas a smaller value reduces the computational cost and approximates the greedy behavior.

When deploying beam search in a practical situation as the one described in Figure 3.4, some issues may appear. For instance, the sequence prediction model provides a probability distribution over the entire object set even though, at a certain step, one of the objects may not be available to pack. The methodology proposed in Section 4.3.2 describes a possible approach to address these issues in practice.

3.5 Statistical Analysis

This section briefly introduces important concepts and tests in statistics that are useful for analyzing experimental results. In particular, the section introduces statistical tests that evaluate the difference of a parameter in two populations, as well as the important concept of power analysis that analyzes the impacts of the experimental design on the validity of ensuing results.

3.5.1 Comparing Two Proportions

The two proportions Z-test is commonly used in statistical analysis to compare the proportion of a parameter in two samples. The goal of this test is to infer if the difference in the sample proportions is due to an actual effect in the populations from which the samples were taken. For instance, this test can be used in a vaccine trial study to compare the proportion of people that were infected by the target disease in the control group and in the treatment group.

Let the proportions of the target parameter in each population be defined as p_1 and p_2. This test evaluates the null hypothesis

$$H_0 : p_1 = p_2 \tag{3.2}$$

against one of the following alternative hypotheses

$$H_1 : p_1 \neq p_2 \tag{3.3}$$

$$H_1 : p_1 > p_2 \tag{3.4}$$

$$H_1 : p_1 < p_2 \, . \tag{3.5}$$

However, as with any Z-test, there is an underlying assumption that the distribution of the test statistic can be approximated as a normal distribution. Since this assumption derives from the Central Limit Theorem, a requirement for the applicability of these statistical tests is that the sample size is sufficiently large.

In cases where the sample size is small, there are alternative statistical tests that can still be considered. One of the most popular is Fisher's exact test [45], so called because it belongs to a subset of tests that make no assumptions about the normality of the test statistic. Instead, these tests calculate the probability of getting a result that is as extreme or more than our sample and hence provide the *exact* test score.

Another alternative is Boschloo's test [46] which has the advantage of more accurately rejecting the null hypothesis when compared to Fisher's test, and thus it is usually a better choice [47]. This test, also part of the exact tests class, is very similar to Fisher's test. In fact, it results from a small modification of the latter that leads to a smaller chance of failing to reject the null hypothesis [46]. Since it derives from Fisher's test, Boschloo's test also essentially determines the likelihood of obtaining an outcome as extreme or more than our sample.

The hypotheses for this test are formulated as described in Equations (3.2) to (3.5) where the choice of which alternative hypothesis is dependent on the characteristics of the problem. The p-value obtained from the test will be compared to the significance level, α, to reject (or not) the null hypothesis. Although

there is no universally accepted standard choice for the significance level, a common choice is to set it to 0.05 as this represents an adequate trade-off between the risk of incorrectly rejecting the null hypothesis and the risk of not detecting a false null hypothesis [48].

3.5.2 Power Analysis

The significance level, α, is an important parameter of any statistical test. Intuitively, it encodes how sure the experimenter can be in rejecting the null hypothesis. It is the probability of a type I error - rejecting the null hypothesis when it is actually true. Thus, one may assume that the lower the significance level the better.

However, there is a flip side to this assumption that is often overlooked, which is the statistical power of a hypothesis test. For instance, most of the research in the field of human-robot interactions fails to perform a power analysis [49]. Statistical power is related to the probability of type II errors - failing to reject the null hypothesis when it is actually false. Statistical power is the probability of *not* incurring a type II error. An underpowered study may report inconclusive results (did not reject the null hypothesis) when in fact the effect is present in the population.

The flip side of lowering the significance level is that the power of an analysis is correlated with it. Thus, reducing the probability of a type I error increases the probability of a type II error. Hence, as highlighted in [49], conducting a power analysis is important to obtain more insights into the validity of the conclusions of a statistical test. Furthermore, such an analysis can also be used prior to collecting the samples to verify if the projected sample size is large enough to obtain sufficient power.

Power calculation is dependent on the hypothesis test, however graphical interfaces are available that automate the process, such as G*Power [50]. It is common to consider a power of 80% or more acceptable and a significance level of 5%, as suggested by Cohen [48].

3.5.3 Effect Size

Even if a test has an appropriate power and significance level, and is able to reject the null hypothesis, its results may still be insignificant. This can occur because a hypothesis test does not test for a *significant* difference, but rather for *any* difference. In other words, statistical tests can correctly identify an effect in the populations but this does not mean that the effect is significant, only that it exists.

Because of this fact, there is a separate measure of significance: the effect size. It measures the significance of a result on a normalized scale that is independent of the quantities in the study. There are multiple measures of this effect size, but a common choice for evaluating the difference between two

proportions is Cohen's H [48]. Given a proportion p, one defines its arcsine transformation, φ, as

$$\varphi = \arcsin(\sqrt{p}). \tag{3.6}$$

The effect size measured by Cohen's H, represented as h, is then given by

$$h = |\varphi_1 - \varphi_2|, \tag{3.7}$$

where φ_1 and φ_2 are the arcsine transformations of the proportions of the target parameter in each population.

Cohen [48] provides the following interpretation of this effect size:

- $h = 0.2$ indicates a small effect size.
- $h = 0.5$ indicates a medium effect size.
- $h = 0.8$ indicates a large effect size.

In conclusion, one can be confident that there is a significant effect when conducting a hypothesis test if the test has an adequate significance level and statistical power, as well as an effect size that reveals an important effect. Even under these conditions, one must not forget that the results from such a statistical analysis are not guaranteed, but rather only likely.

4

Results

Contents

This chapter is divided into three main sections. The first one presents the virtual environment used for the packing demonstrations and the technical details of the data that we collect from it. The second section analyses the actual data, for instance, the number of participants and general statistics. Finally, the last section presents how we use these data to predict the packing sequence for a set of objects.

4.1 Real Data From a Virtual World

This section describes the virtual environment that we created to collect the demonstrations of human experts performing the bin packing task. Furthermore, we discuss its limitations and detail what data are collected and how they are stored.

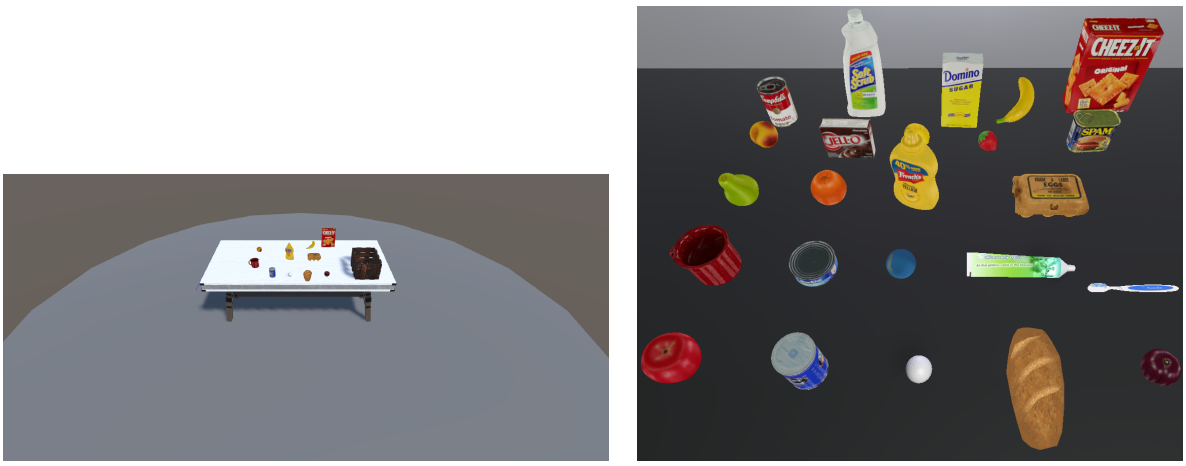
4.1.1 The Virtual Environment

Due to the advantages highlighted in Section 3.1, we create a virtual environment in Unity [39], illustrated in Figure 4.1(a), to collect data from humans performing the bin packing task. This environment consists of a circular area where the user is free to move and a table in its center. The table is where the task takes place: the user should pack the objects on the table into a box.

For each demonstration[1], the objects are a random subset of the entire object collection illustrated in Figure 4.1(b), generated such that the total sum of each object's bounding box volume is between 70% to 90% of the box volume. This is to ensure that the task is not trivially solved, forcing the user to carefully plan the placement of each object. In some scenes we allow repeated objects whereas in others we enforce that there is, at most, one instance of each object. This ensures that the dataset contains examples of all possible scenarios. At the start of each demonstration, all objects are spread out on the table to ensure that the participant can see and grasp all objects. Out of the 24 different objects, the majority were chosen from the YCB dataset [51] and the others were obtained from public object model platforms.

The participants interact with the virtual world via the physical controller. The position and orientation of the controller are mimicked by the virtual gripper and a pressure-sensing button controls the closing and opening of the virtual gripper's fingers, as shown in Figure 4.2. The objects are configured as rigid bodies with colliders, which means that they are affected by gravity and friction, and can collide with other objects.

[1] A video of the author of this thesis performing the packing task can be found here https://youtu.be/TUd-eCDG5i8



(a) (b)

Figure 4.1: (a) Virtual environment created in Unity for data collection. (b) Collection of objects used in the experiments.

(a) (b)

Figure 4.2: Illustration of how the virtual gripper, shown in (b), is operated with the physical controller shown in (a).

When a participant grasps an object it is rigidly attached to the gripper and the relative pose between the gripper and the object is maintained during the manipulation until the object is released. Ideally, this would not occur and instead, the grasping would be based on the friction between the gripper and the object, and on the forces that are being applied. However, all approaches to achieve such a physics-based grasping with Unity's simplified friction physics failed during the development of the environment.

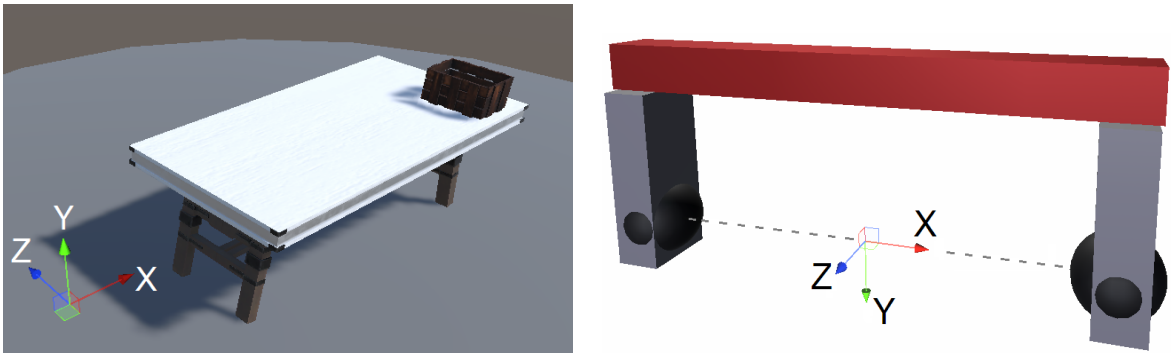
Another limitation of the physics in the virtual environment which we also addressed is the lack of feedback of forces and contacts. Since the user does not feel contact forces when the gripper collides with an object, it is more difficult to accurately perceive if the gripper will hit an object or the box during a manipulation. Consequently, we verified that the participants would accidentally collide with objects that they had already packed into the box when placing another object. Often this would lead to the

displacement of the objects inside the box. To prevent this, we give haptic feedback through the physical controller whenever the virtual gripper collides with an element in the virtual environment. Furthermore, we also disable collisions between the gripper and the objects when the user is packing an object inside the container. This helps ensure that the objects stay in their original positions inside the box after being packed.

Before each demonstration, the participants are shown how to operate the physical controller, the purpose of the task, and were asked to behave as realistically as possible. Particular emphasis was given to considering the objects' fragilities since this property is not present in the virtual environment but must be considered when stacking objects on top of each other. For instance, the bleach cleanser, a hard-plastic bottle, is less fragile than a small strawberry.

4.1.2 The Data

Since the data includes 6-DoF poses of the gripper and objects, one must first establish the relevant world and local reference frames. The world reference frame is oriented according to Unity's standards: the Y axis points up and the X and Z axes are an orthogonal basis of the ground plane, as illustrated in Figure 4.3(a). The gripper reference frame is presented in Figure 4.3(b). Furthermore, each object has its own local reference frame which is defined by its mesh file and the placement box also has a reference frame with normalized coordinates, as illustrated in Figure 4.4.



(a) **(b)**

Figure 4.3: (a) Virtual world reference frame. (b) Gripper reference frame.

All data is saved with the JSON format [52] (apart from images), to enable cross-platform and cross-language compatibility. Each participant is attributed a unique integer identifier and performs the packing task for multiple scenes. For each scene, the following data is recorded and structured in separate files, as summarized in Table 4.1.

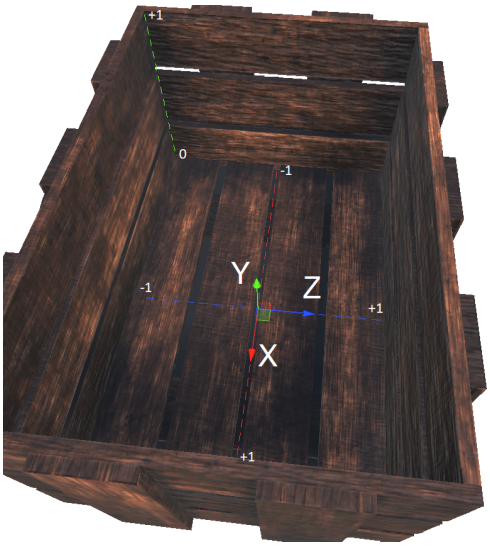


Figure 4.4: Reference frame of the box used to pack objects. The coordinates are normalized.

Table 4.1: Overview of the contents of the dataset and its file structure.

Parameter	File name	Format	Description
Grasp Pose	PickPlace_dataset	JSON	6-DoF grasp pose
Placement Pose	PickPlace_dataset	JSON	6-DoF placement pose inside the box
Packing Sequence	PickPlace_dataset	JSON	Sequence in which the objects were packed
Object Trajectory	<obj_id>_trajectory	JSON	Trajectory of the object from the table to the box, sampled at 20 Hz
Headset Trajectory	main_camera_trajectory	JSON	Trajectory of the headset during the experiment, sampled at 20 Hz
Objects in Scene	initial_objects	JSON	List of objects and their poses at the start of each scene
Top-Down View	top_down	PNG	Top-down image of the initial layout of the objects

The file PickPlace_dataset.json contains the sequence in which the objects were picked up from the table and placed inside the box, as well as the corresponding grasp poses. It has the following fields:

- Object identifier: the object's name, number, and unique ID (needed to distinguish between multiple copies of the same object). Eg: 003 cracker box-01234

- Pick translation: a 3-dimensional vector with the translation from the object's reference frame to the gripper reference frame, upon grasping the object.
- Pick rotation: a 3x3 rotation matrix that represents the orientation of the gripper w.r.t. the object's reference frame, upon grasping the object.
- Place translation: a 3-dimensional vector with the coordinates of where the object was placed inside the box, normalized to $[-1, 1]$ for the width and length dimensions and $[0, 1]$ for the height dimension, according to the reference frame in Figure 4.4.
- Place rotation: a 3x3 rotation matrix that represents the orientation in which the object was placed inside the box, w.r.t. the world reference frame.

An example of a grasp pose saved in the dataset is presented in Figure 4.5.

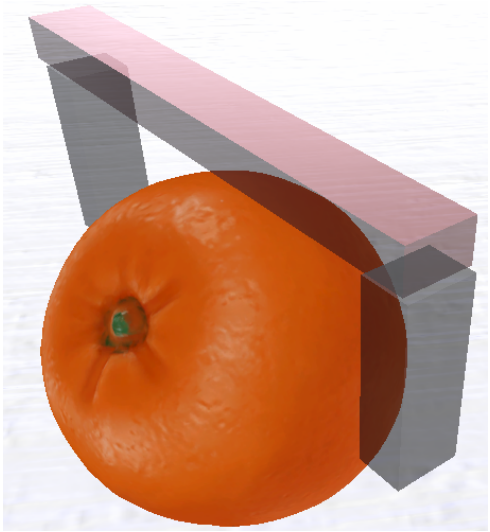


Figure 4.5: An illustration of a grasp pose (both translation and rotation). The semi-transparent gripper was positioned according to the collected data.

The file <object_id>_trajectory.json (where <object_id>is the object's identifier) contains the trajectory of the corresponding object, as visualized in Figure 4.6. The trajectories are sampled at 20 Hz for computational efficiency and also because practical tests showed that 20 Hz is a sufficient sampling frequency for human motion in VR. This file contains the following parameters for each sampled pose:

- Timestamp of the sample.
- Object translation: a 3-dimensional vector with the translation from the world reference frame to the object's reference frame.
- Object rotation: a 3x3 rotation matrix that represents the orientation of the object w.r.t. the world reference frame.

The file **top-down.png** is a top-down view of the initial layout of the objects on the table, captured from a secondary camera. An example image is presented in Figure 4.7.

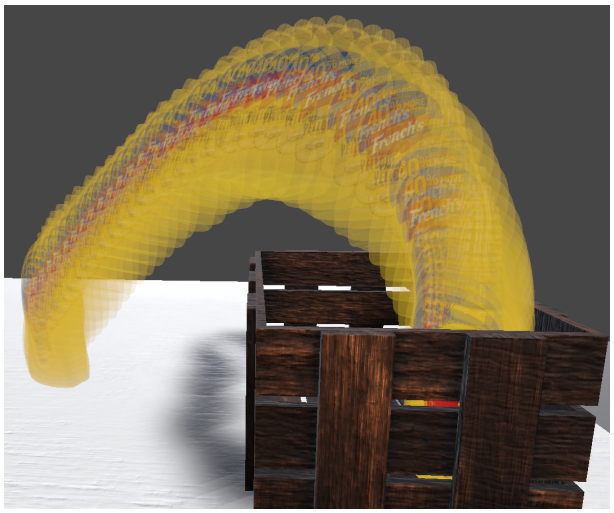


Figure 4.6: An example object trajectory recorded during a demonstration. One semi-transparent copy of the target object was spawned at each recorded pose.

Figure 4.7: An example top-down view of the initial layout of the objects on the table.

The reason for recording all these variables is that, if necessary, it would be possible to replay each experiment and record additional parameters or different representations of the data. In other words, by recording all the essential task parameters we ensure that each demonstration can be accurately replayed to extract different parameters or representations, such as a top-down or first-person video of the demonstrations. Inevitably, the dataset does not capture all possible parameters of the task. For instance, the dataset does not collect the participant's eye gaze or grasping force.

4.1.3 Assumptions and Limitations

Although this dataset attempts to be as general-purpose as possible, some underlying assumptions

must be clarified. Some of these are derived from the goal of the thesis while others are simplifying assumptions so that the dataset could be collected within a limited period of time. This dataset assumes that:

- All items are initially spread out on the table with no overlap. This is to ensure that all items can be picked up in any order since the goal is to learn the packing sequence, which would be influenced by overlaps and unreachable objects.
- All items can be grasped with the virtual gripper in, at least, one grasping pose. This means that larger items, such as a large cube-like object, are not considered.
- The participants can accurately estimate the fragility of an object from observation only since this property cannot be simulated in VR. This is a reasonable assumption since all the objects are common objects that the participants were already familiar with.

Furthermore, Unity's physics engine introduces some limitations as it is optimized for speed and not for the most realistic behavior (it is primarily a game engine and not a physics simulator, hence the trade-off). As such, there are some aspects that should be considered when using these data.

Firstly, simulation data is always different from real data [53]. Depending on the type of data, this can be an important factor to consider. In the data we collect, this Simulation-to-Reality (Sim2Real) gap is most evident in the unrealistic appearance of the objects. This would be a limiting factor for directly using the images obtained in VR to train a neural network, for instance. However this is not a critical issue since we collect all the necessary parameters to replicate a demonstration in a more realistic simulator, such as Blender [54]. This would allow the collection of photo-real images of any portion of the demonstrations.

Secondly, there is also the possibility of unnatural object manipulation due to the lack of feedback to the user. In other words, the lack of feedback on weight, texture, and other important physical properties may affect how the participants manipulate objects. However, during the experiments the participants reported that the virtual reality interface was realistic and intuitive to use. As such, the object manipulations should mostly resemble real object manipulations.

Finally, Unity's approximate collision handling system may introduce artifacts in the interactions between objects. As reviewed in Section 3.1, Unity uses colliders to detect collisions between objects and not the actual object mesh. To recall, these colliders can have one of a few primitive shapes (box, sphere, capsule) or they can be a *convex mesh collider* which approximates objects with a convex polygonal hull. This not only limits the types of objects that can be used but can also lead to an imperfect detection of the grasp width, which is the distance between the gripper's fingers upon grasping an object. This occurs because contact points are detected between the gripper and the object's colliders, not the actual object mesh. An example of the approximation considered by an object's colliders is shown in Figure 4.8. Because of these differences, we do not record the grasp width since, for some objects, it

would not correspond to the real grasp width.

Figure 4.8: An example object and its convex mesh collider (in green) for which there is a clear difference between the actual object shape and the approximated object collider.

4.2 Unpacking the Dataset

This section presents a statistical analysis of the collected data[2]. More specifically, the section includes general statistics on the participants, details about the duration of each experiment and its difficulty, and an analysis of particular aspects of the task such as placement poses.

4.2.1 General Statistics and Task Duration

The experiments were conducted using an HTC Vive Pro headset, totaling 75 experiments and 43 different participants (some participants volunteered for the experiment more than once, hence why the number of experiments is larger than the number of participants). Their ages were approximately comprised in the interval of 20 to 65 years old. Recall that in each experiment the participant repeats the task multiple times and that for simplicity, we denote each box packing as a *scene*. In total the dataset contains 263 scenes, or in other words, 263 boxes were packed by the participants. This corresponds to 4644 grasp poses and object placements, which on average corresponds to 194 manipulations per object.

[2]The dataset can be accessed here https://vislab.isr.tecnico.ulisboa.pt/datasets_and_resources/#BoxED

By plotting a histogram of each scene's duration, presented in Figure 4.9(a), we observe that it approximates a gamma distribution with an average duration of 1 minute and 17 seconds, and a standard deviation of 36 seconds. Furthermore, a plot of each scene's duration as a function of the corresponding number of objects in it is presented in Figure 4.9(b). The pink line in this figure was obtained with a standard least-squares linear regression and indicates that, on average, each object adds 4 seconds to the task's duration. This result is according to what was expected since the difficulty of the task naturally increases with the number of objects that need to be packed.

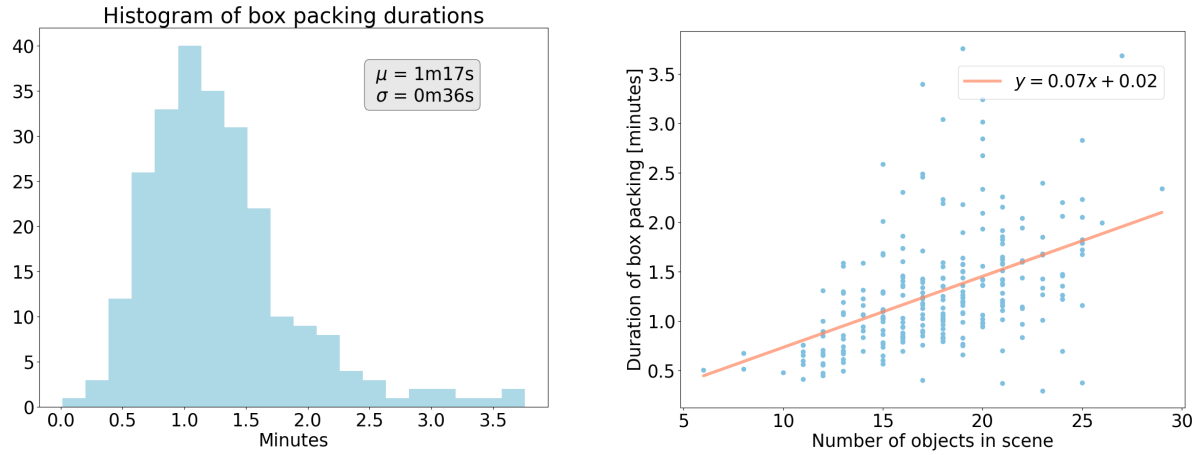


Figure 4.9: (a) Histogram of all scenes' durations. (b) Duration of each scene as a function of the number of objects. The pink line was obtained with a linear regression.

4.2.2 Placement Poses

Analyzing the placement poses is not trivial since they are six-dimensional variables. Thus, for better visualization the following analysis considers separately its two components: position and orientation.

To detect frequent patterns in the positioning of different objects inside the box, Figure 4.11 shows a plot of the placement positions for three different objects, in a top-down view. In other words, this plot ignores the height of the placement and considers only the horizontal coordinates inside the box. The three objects are a strawberry, a pudding box (a medium-size cardboard box), and a bleach cleanser, all shown in Figure 4.10.

Figure 4.11(a) reveals that the strawberry is placed uniformly throughout the box, however the average placement is closer to the side of the box closest to the participant. This occurs because the strawberry is one of the most fragile items and so participants tended to pack it only after all other objects, confirming that the participants considered the objects' fragilities during the task. In contrast, both

the pudding box and the bleach cleanser have a very uneven distribution. It is clear that they are generally placed along the edges of the box and further away from the participant. This is likely a strategy that most participants intuitively adopted due to two factors. Firstly, to optimize the available space inside the box, and secondly because large objects would become an obstacle for the task if they were placed closer to the participant since they would obstruct subsequent placements.

Figure 4.10: Objects considered in the placement pose analysis: (a) strawberry, (b) pudding box, and (c) bleach cleanser.

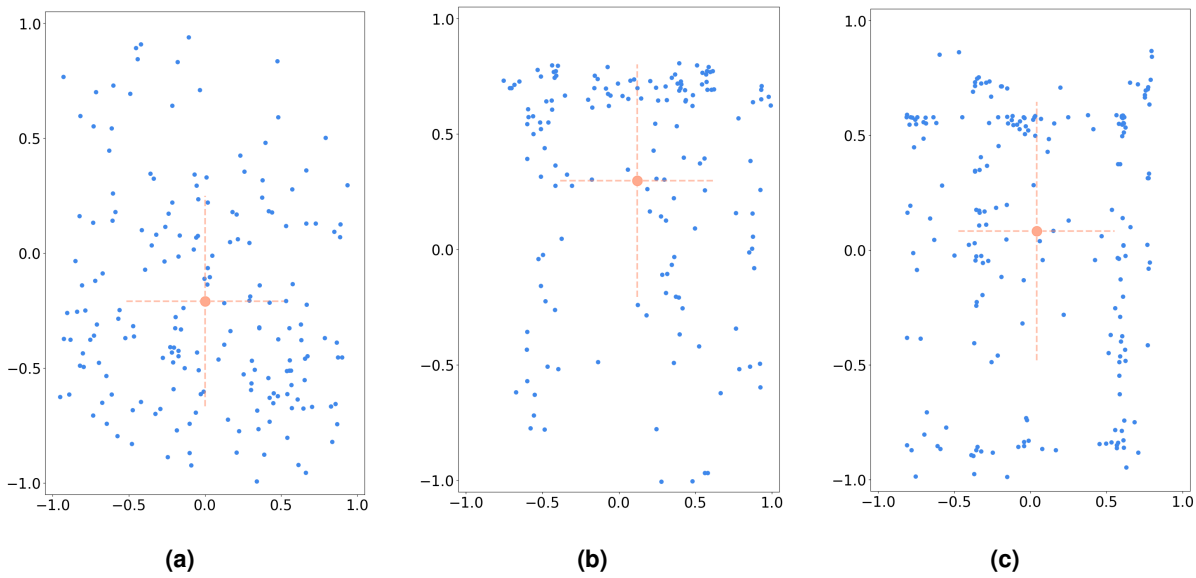


Figure 4.11: Placement positions inside the box of three objects, from a top-down view. The larger pink circle is the average position and each dashed line is the standard deviation along the corresponding axis. The objects are (a) strawberry, (b) pudding box, and (c) bleach cleanser.

A similar strategy can be used to analyze the orientation, except in this case, the three angles that

compose it - the rotations around each axis - are visualized independently of the others. This analysis is conducted for the same three objects and presented in Figure 4.12.

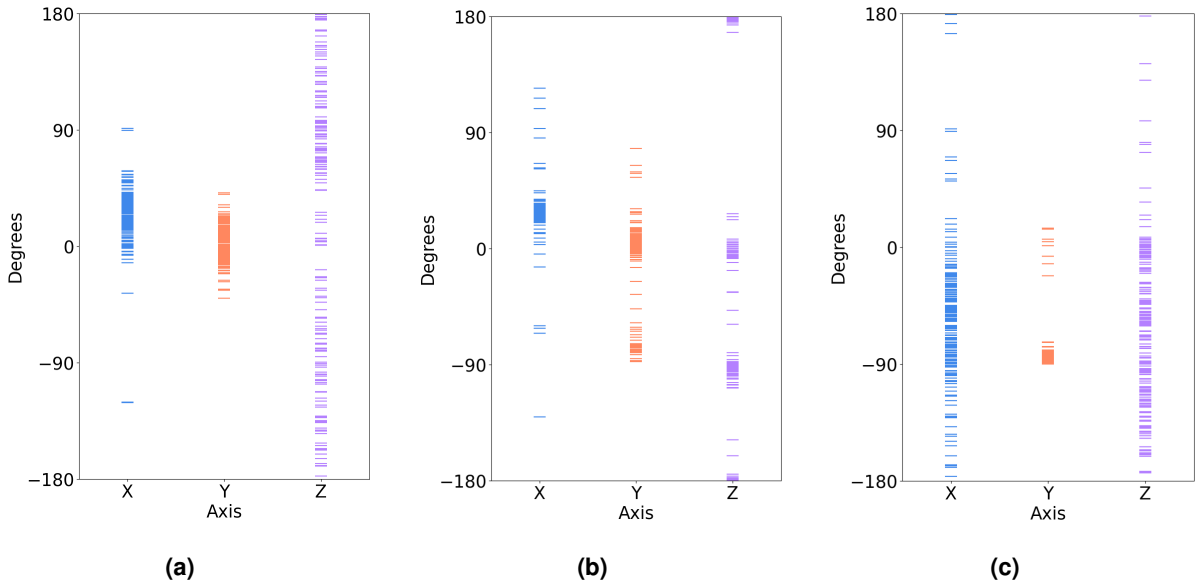


Figure 4.12: Placement orientations inside the box of three objects, where each line segment corresponds to one placement. The objects are (a) strawberry, (b) pudding box, and (c) bleach cleaner.

The visualization presented in Figure 4.12 reveals, once again, distinctive patterns for each object. For instance, the strawberry's placement rotations shown in Figure 4.12(a) have near uniform distribution around the Z axis. This is a consequence of the strawberry's lack of symmetry around this axis, which consequently means that it can be safely placed in any orientation around it. On the other hand, the bleach cleanser presents a very skewed distribution around the Y axis (which in Unity points up). This is a consequence of its symmetric shape which leads participants to frequently place it aligned with the edges of the box.

This analysis of the placement poses reemphasizes the vast implicit knowledge humans use to pack a box with groceries and motivates our approach of learning these patterns from the data rather than relying on hand-designed rules or constraints.

4.2.3 Object Packing Sequences

Another differentiating factor of BoxED is that it contains 263 packing sequences. Each of these is an ordered sequence of object names that describe the order in which the objects were placed inside the box. For instance, the sequence

$$bleach\ cleanser \rightarrow sugar\ box \rightarrow bread \rightarrow toothpaste$$

indicates that 4 objects were packed, starting with the bleach cleanser and ending with the toothpaste.

This order depends on multiple factors, including the objects that need to be packed and how they are placed inside the box. This last factor refers to how the sequence will change depending on whether the participants pack the box in an orderly manner (i.e., starting on one side of the box and progressively placing objects until the box is full) or just place the objects randomly inside the box. The latter scenario would render these sequences mostly useless since they would contain no information about the final layout of the objects inside the box.

To prevent this, the participants were asked to pack the box orderly from one side of the box until reaching the other. The intuition behind this request is that if an object A, obj_A, is placed after another object B, obj_B, then it is likely that obj_A is stacked on top of obj_B, and consequently obj_A is likely more fragile than obj_B. This means that from modeling the sequences alone one can extract some information on how to place the objects, as we demonstrate in the next section.

4.3 Predicting the Packing Sequence

This section describes how we learn to predict the packing sequences from our human demonstrations dataset. Furthermore, this section will also describe how we test and validate our model.

4.3.1 Learning the Packing Sequence Prediction Model

As highlighted in Section 2.1, many methods exploit Markov chains and their extension, Markov Decision Processes, as the core of their strategies to learn from demonstrations. The popularity of this model stems from its simplicity and from not requiring large amounts of data to train. Generally, the state encodes relevant information about the target environment and agent (a game, a robotic arm, etc.) at each instant, and the agent's actions cause state transitions. Here we show how a Markov chain can be used to model the packing sequences.

When attempting to model the packing sequences in BoxED, a logical first step is to apply a sequential pattern mining algorithm. Such an algorithm will extract frequent patterns from the packing sequences and their associated probabilities. For simplicity, we can consider only the patterns with two objects. An example of one of these outputs could be the pattern $cracker\ box \rightarrow sugar\ box$, along with the associated conditional probability, $p(sugar\ box|cracker\ box)$. An interesting remark appears when this output is compared to the formulation of a Markov chain: they are the same.

The patterns and associated probabilities mined from the packing sequences naturally formulate a Markov chain when joined together. This chain's complexity is limited due to the assumptions that underlie this formulation:

1. The choice of what object to pack next depends only on the last object. This is a consequence of only considering the mined patterns that have two objects.
2. The aforementioned choice does not depend on the pose of the last object. This derives from analyzing the packing sequences independently from the placement poses.

Also, note that this formulation ignores the restrictions imposed by what objects are still available to pack. However, as we will show later in this section, this issue can be addressed when sampling the Markov chain. As a consequence of these assumptions, the state is reduced to the name of the last packed object and transitioning to a new state corresponds to packing another object. To the reader this may seem an oversimplification of the problem. In fact, when deciding what object to pack next, an individual will likely take into account the last few objects that were packed (not just the last one), their disposition inside the box, and the objects that remain unpacked. Yet we show in the remaining of this section that accurate results can be obtained with this formulation.

To formalize this model, let $O = \{obj_A, obj_B, ...\}$ be the set of all objects. We define $S = \{<start>, O\}$ as the set of all possible states, where at each instant the state indicates what object was packed last, and the state $<start>$ indicates that no object was packed inside the box yet. Each transition probability, from an object obj_A to obj_B, is represented by the conditional $P(obj_B \mid obj_A)$ and indicates the likelihood of placing obj_B after placing obj_A.

Furthermore, we need to estimate the transition probabilities. Recall from Section 4.2.3 that participants were asked to pack the box orderly, such that if an object ”A” is placed after object ”B”, then ”A” is likely stacked on top of ”B”, or at least is next to it. However this is not true for all pairs of objects in the sequences, and so we need to extract only the meaningful patterns in the dataset.

This is precisely why SPM algorithms are useful for this task: they enable extracting meaningful patterns from the sequences. We extract the pairs of objects that appear in the sequences with a frequency above an adjustable threshold similarly to the first two iterations of the algorithms described in Section 3.2: first analyze and prune individual objects, and then repeat for pairs of objects. At the end of each step, the sets of objects whose support (probability of appearing in a sequence in the dataset) is smaller than the threshold are discarded. For instance, if the pair $strawberry \rightarrow tuna\ can$ appears in only 1% of all sequences, then it is not significant and can be discarded.

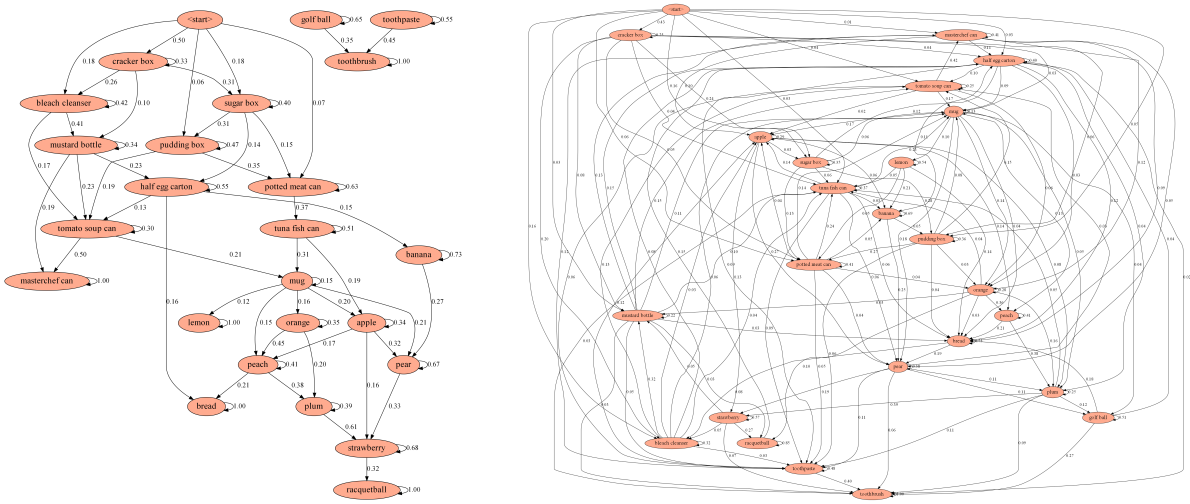
The adjustable threshold controls how sensitive the model is to the transitions in the dataset. A higher threshold retains only the most frequent transitions whereas a lower threshold retains more transitions and produces a Markov chain that is more densely connected, at the cost of including transitions that may be irrelevant. This trade-off is illustrated in Figure 4.13.

The impact of the threshold is clear and can even lead to multiple disconnected chains as seen in Figure 4.13(a). This threshold was adjusted to satisfy the following guidelines:

1. It should be small enough to include at least one transition for each object in the object set O.

Otherwise, the sequence prediction would fail to pack all objects.

2. It should be large enough to minimize loops in the chain, in order to maintain a hierarchical top-down structure.



(a) **(b)**

Figure 4.13: Influence of the minimum frequency threshold on the number of transitions extracted from the data. In these examples, the thresholds are (a) 0.01 or 1%, and (b) 0.05 or 5%. Note that the text in these figures is small since the purpose is to analyze the shape of the chains and not individual nodes.

After a trial-and-error analysis, a threshold value of 0.032, or 3.2%, satisfied these criteria. One final minor adjustment is necessary: since we only select the frequent pairs of objects, some transitions are discarded and thus we need to normalize the remaining transitions' probabilities. This ensures that the total sum of the probabilities of the transitions starting from each node is 1.

Having obtained the relevant transitions and their normalized probabilities, we can instantiate our Markov model. It could be visualized as a simple table of transition probabilities between different states. However, this would not be very intuitive. Instead, using the graph visualization toolbox PyGraphviz [55] we are able to plot the complete Markov chain in an easily interpretable format. The resulting Markov chain is presented in Figure 4.14.

Remarkably, this Markov chain captures precisely what was desired even though we considered simplifying assumptions. Larger and more robust objects are placed first, followed by smaller and more fragile objects. For instance, the topmost object is the cracker box, which is the largest object in the dataset and was usually the first one to be placed inside the box by the participants. On the other hand, the bread and the toothbrush are the bottommost objects since they are the most fragile and smallest, respectively.

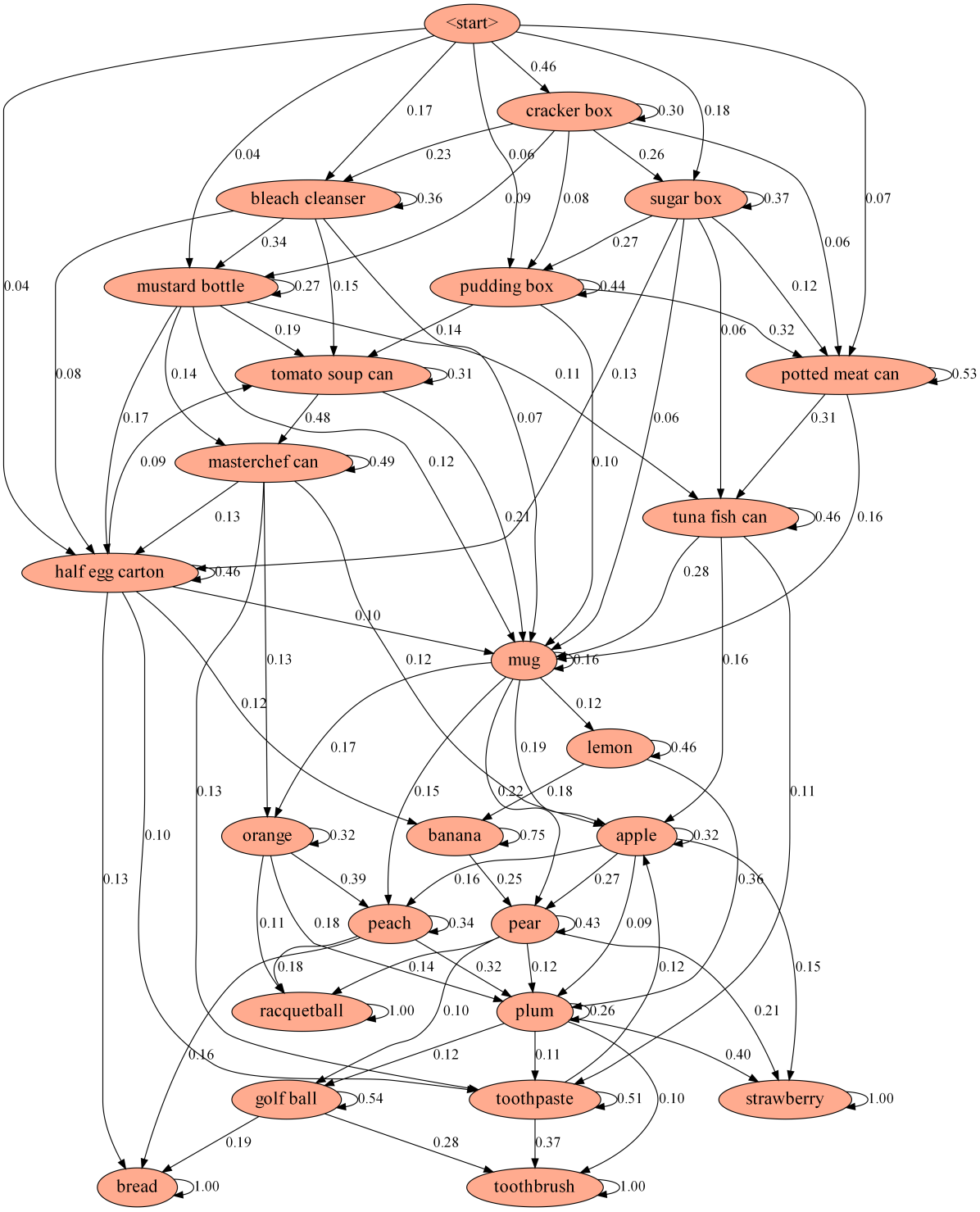


Figure 4.14: A Markov chain modeling the object packing sequence. Each node corresponds to a state and the numbers on the transition arrows are the corresponding probabilities. This chain was obtained with a threshold value of 0.032 or 3.2%.

Furthermore, all transitions correspond to objects that could be stacked, one on top of the other, safely and stably, or correspond to pairs of objects that were frequently placed together, side-by-side. The chain does not contain illogical transitions, such as placing a heavy object on top of a fragile object (for instance, $strawberry \rightarrow tuna\ can$). Finally, this plot shows that the sensitivity threshold in the pattern mining step was successfully adjusted such that all objects are included in the model and there are very few loops.

4.3.2 Predicting a Packing Sequence with a Modified Beam Search

Regardless of how well the Markov model captures the packing patterns and strategies that humans use, it is of no use without an adequate sampling mechanism. As such, this section discusses how we generate a packing sequence prediction from our model.

As discussed in Section 3.4, when sampling from stochastic models beam search generally provides better sampling than naive approaches such as greedy search. Other strategies as epsilon greedy search could also be deployed to introduce more fluidity or randomness to our rigid model. However, we choose to test beam search as our first approach. As the title of this section indicates, we don't use a standard beam search, rather we introduce some modifications to adjust it to models such as ours.

Recall that in each scene the set of objects that need to be packed is a random subset of all objects, and consequently, some objects in the Markov chain may be unavailable. This is different from the text generation task, for instance, where the models output a probability distribution over a fixed dictionary of words at each step. In this case, standard beam search cannot be applied since the search must be restricted to only the objects that weren't packed yet. This not only restricts what child nodes can be considered during the search, but it also creates the peculiar situation illustrated in Figure 4.15.

If a node has no possible child nodes for a set of objects that were not yet packed, it may still have a transition to a valid grandchild node. By grandchild node we denote all nodes reachable from the current node with two or more transitions. For instance, during the search the node $bleach\ cleanser$ may have no valid children but it may have a valid transition to the $masterchef\ can$ through the $tomato\ soup\ can$. As a consequence, when sampling a sequence the algorithm must search all the child and grandchild nodes of the current node.

This can make the search process very slow because the algorithm would need to search through a large portion of the chain to find valid transitions for each node. As such a modification is needed to optimize the sampling algorithm. The key observation of our solution is that the Markov chain is static - it never changes. Because of this, we can build a table of all possible transitions for each object before executing the algorithm. These tables can then be stored along with the Markov chain.

For each state in S (the set of all states), we store all possible transitions in levels. The first level con-

tains the transitions to direct child nodes - those that require only one step. The second level contains the transitions to the grandchild nodes that require two steps, and so forth. For instance, considering Figure 4.15, the $bleach\ cleanser$ node would have a table with the $mustard\ bottle$ and the $tomato\ soup\ can$ in the first level, and the $masterchef\ can$ in the second level. This is essentially a more detailed (and redundant) description of our Markov chain. The original chain contains only the transitions to the child nodes for each state - the minimum amount of information needed to store the chain. This new, multi-table representation contains all the transitions for all states. This increases the algorithm's memory complexity but makes it much faster.

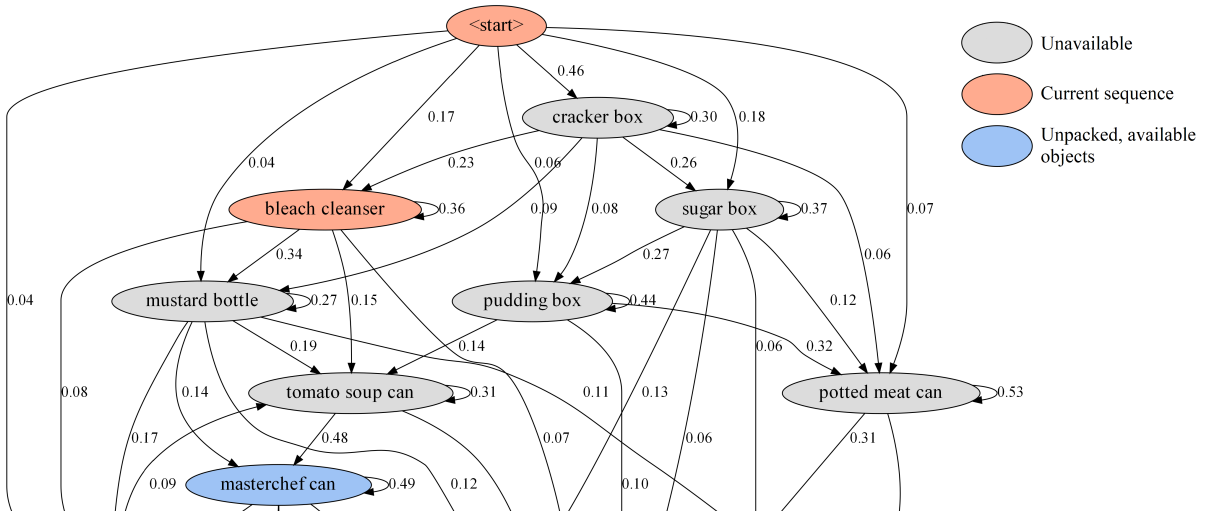


Figure 4.15: Example situation that illustrates that even when all child nodes are invalid, there may still be a valid transition that requires more than one step.

Furthermore, one more change to the standard beam search is required. During the search process we consider that a sequence is complete when it reaches a leaf node - a node that has no more valid transitions. But consider the following situation: as the beam search progresses with a beam width of 2, for simplicity, the state of both beams - called $Beam_A$ and $Beam_B$ - is

$$Beam_A: <start> \rightarrow\ cracker\ box\ \rightarrow\ sugar\ box$$
$$Beam_B: <start> \rightarrow\ bleach\ cleanser\ \rightarrow\ mustard\ bottle,$$

whose respective likelihoods are, approximately, 0.12 and 0.06. Consider also that besides the objects in each beam, there is only one more object left to pack: a $potted\ meat\ can$. In the following search iteration, this object can be added to $Beam_A$ reducing its likelihood to 0.01, but not to $Beam_B$ as there is no transition from the $mustard\ bottle$ to it.

If, at the end of this iteration, the algorithm needs to discard one of these updated beams (to replace it with a better option), which one should be discarded? Both beams are valid predictions but if the search considers only the likelihood of a sequence then it will inevitably be biased toward shorter sequences

that have higher probabilities. This effect, in conjunction with the first change we made, would encourage the algorithm to find sequences that reach a leaf node with the smallest length possible.

Thus, we must introduce a preference for longer sequences to the detriment of complete sequences. In practice this can be accomplished with two additions:

1. During the search process, choose sequences that can still expand to more nodes to the detriment of sequences that have already reached a leaf node.
2. Enforce that transitions to a new node are as small as possible. In other words, start by searching for a valid state in the first level of our new representation. If none are found, proceed to the second level, and so forth.

The final search algorithm is presented in Algorithm 4.1.

Algorithm 4.1: Modified beam search algorithm used to sample a sequence prediction from our model.

```
1  beam_width ← 5;
2  transitions ← load level representation of Markov chain;
3  unpacked ← list of unpacked objects ;        /* Obtained from the virtual environment */
4  open_list ← initial beam with <start> state ;        /* Contains active search beams */
5  while not exhausted all transitions for all beams in open_list do
6      cur_transition_level ← 1 ;                        /* Transition level to search in */
7      while at least one beam in open_list has transitions for cur_transition_level do
8          foreach beam in open_list do
9              if transitions has valid child states for this beam and cur_transition_level then
10                 open_list +=
                     GenerateNewBeams(beam, transitions[beam][cur_transition_level]);
11                 remove beam from open_list;
12         if at least one beam was extended then
13             sort open_list according to beam probability;
14             keep top beam_width beams and delete the others;
15             break;
16         else
17             cur_transition_level ++;
18 return most likely beam in open_list ;    /* All valid transitions have been exhausted */
```

In this implementation we defined the beam width to be 5, meaning that the search maintains the top 5 most likely beams between each iteration. In line 2 we load the new representation of the model, which contains all possible transitions for each state, sorted by level number. The algorithm will exhaust all possible transitions before returning the most likely sequence. In each iteration beginning at line 6, the algorithm will iteratively search, for each beam and for the current transition level, if there are valid transitions, prioritizing small step transitions. In lines 12-15 the standard beam search is implemented.

4.3.3 Evaluating the Model's Performance

Now that an appropriate sampling mechanism has been implemented, this section focuses on describing how to test the sampled sequences. The main challenge lies in the fact that, for a given set of objects, there are numerous valid packing sequences. Depending on how the objects are placed inside the box, two distinct sequences can lead to an equally efficient packing. This dependency on object placement makes evaluating a sequence very difficult without actually placing the objects inside the box.

Ideally, we would use a method to predict an adequate pose for each object and carry out the box packing, either in simulation or with a robot. In the end, a measure of the sequence prediction's performance could be how efficiently the objects had been packed inside the box. However, currently no method is able to match a human's efficiency at packing objects in a container. Furthermore, if the sequence prediction model is learned from human behavior, then the placement pose prediction should also be human-like to ensure compatibility between the two models. Thus instead of a placement pose prediction method, we decide to have a human pack the objects in the box.

Drawing some inspiration from the Turing test in which a human evaluator must distinguish between human or computer generated text responses, we develop a new virtual environment to test the sequence prediction model. This new environment is visually identical to the environment used to collect BoxED, shown in Figure 4.1(a). The objects still appear spread out on top of the table, the participants interact with the environment as before, and the goal is the same: pack all objects inside the box.

However, the participant no longer chooses the order in which the objects are packed. Instead, the sequence is indicated by highlighting what object should be placed next. Some of the sequences are real sequences that we collected in the first part of our work whereas the others are generated by the computer. At the end of each scene (i.e., after packing all objects), the experimenter asks the participant if the sequence was a real sequence executed by a human or a computer generated sequence. The participants answer "real sequence" if they felt that the sequence was human-like and logical, or "computer generated" otherwise.

There are four types of sequences:

1. Real: sequences generated by participants during the data-collecting experiments. These are randomly selected from BoxED.
2. Random: given the subset of objects on the table, a packing sequence is randomly generated.
3. Beam-N: sample a sequence prediction using the modified beam search algorithm described in the previous section, Section 4.3.2. In this case, no limit is imposed on the length of a sequence prediction.
4. Beam-3: the same as Beam-N except that a maximum sequence length of 3 is imposed.

Each one of the four types of sequences has its purpose. The real sequences are used to verify

that the participants can detect sequences executed by other humans. This tests whether personal packing preferences are sufficiently strong to influence the final evaluation, or if there are general packing strategies that most humans adhere to.

The random sequences are used to answer the question ”Can participants distinguish between logical and random sequences?”. The results from evaluating this type of sequence and from the real sequences will measure if humans are good evaluators for this task and also validate the experimental design. Finally, the results from the Beam-N and Beam-3 sequences will be used to test if our Markov chain and sampling algorithm were able to ”trick” participants into evaluating the sequence as generated by another human. This will prove whether the model captured the general principles that guide individuals in this task.

Note that, a sequence prediction sampled with the Beam-N or Beam-3 strategies may not include all the objects that need to be packed. When this occurs the virtual environment waits until the participant has packed all the objects in the current predicted sequence and then requests a new sequence prediction for the remaining objects. The sampling process always starts from the top of the Markov chain. This cycle repeats until all the objects have been packed.

A new packing sequence prediction is also requested when a participant places an object near the top of the box. We call this latter feature ”stack detection” and the intuition is that if the last object was placed on top of a stack of objects, near the top of the box, the next object will be placed next to this stack, on the bottom of the box. Consequently, this needs to be a large and/or heavy object, which is found at the top of the model and at the beginning of a sequence prediction. This feature emulates the behavior of the participants when we were collecting data: they packed the objects orderly, usually forming stacks of objects until reaching the top of the box.

The Beam-3 variation was created after receiving some feedback from the participants who stated that the Beam-N sequences packed small and fragile objects too soon in the process when there were still larger objects left to pack. This is a consequence of Beam-N sampling a long sequence from the entire Markov model, and consequently, the Beam-3 strategy was developed to contradict this effect. Since sampled sequences always begin at the top of the chain, limiting their length to 3 objects will first place all the objects at the top first and only then place objects further down the Markov chain.

One final remark is due. The previous explanation mentioned that the virtual environment requests a packing sequence prediction. The request is a message sent from the C# script that controls the virtual environment in Unity to a Python server that runs concurrently. The request includes the set of objects that were not packed yet and which prediction method should be used. The response from the server contains the predicted packing sequence. The complete process of requesting a new prediction and receiving the response lasts less than 0.1 seconds since our model can predict a packing sequence in a very short time.

4.3.4 Experimental Results

This section presents and analyses the experimental results obtained from testing the experiment with 29 participants. Firstly, we introduce and plot the results and then proceed to conduct a statistical analysis that quantifies their validity and significance.

In total, we collected 96 evaluations from the participants, averaging 24 per sequence type. The evaluations of the real sequences are summarized in Figure 4.16.

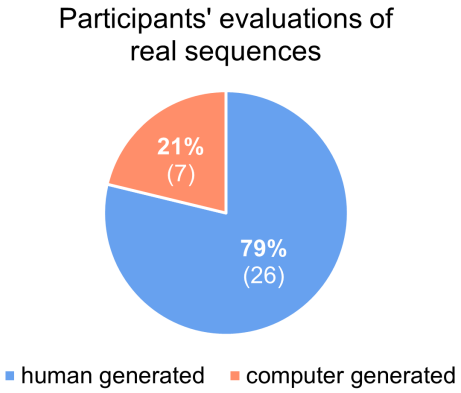


Figure 4.16: Distribution of the participants' evaluations when presented with a real, human-generated sequence.

Overall the result is positive with 79% of the evaluations correctly identifying these as *human generated*. Thus we can infer that participants are generally capable of identifying a sequence produced by another human. However, at 21% the amount of incorrect answers is larger than expected. We suggest that there are two main factors contributing to this error. Firstly are the individual preferences of how to best pack objects in a box which may lead individuals to dislike or deem as illogical another person's packing sequence. The second factor is more arbitrary and is related to the fact that a participant may place a few of the initial objects in positions that become an obstacle or that are incompatible with the rest of the sequence. Nevertheless, the result is positive as the majority of the participants correctly identified the sequence type.

Proceeding to the random sequences, 93% of the participants correctly classified them as "computer generated", as shown in Figure 4.17.

As the vast majority correctly identified that this type of sequence was illogical, we can conclude that there are significant differences between the real sequences and randomly packing objects. In conjunction with the previous result, these data validate the experimental design. This is, humans are good evaluators of this task and can recognize packing strategies in sequences produced by other humans. This implies that there are, in fact, general rules or strategies that most humans follow when packing objects in a box.

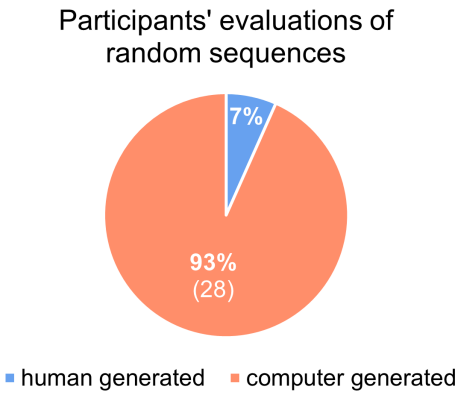


Figure 4.17: Distribution of the participants' evaluations when presented with a random sequence.

We now turn our attention to evaluating whether our model was able to capture some of these general principles that humans consider in this task. Figure 4.18 presents the results for sequences generated from our model and sampled with the Beam-N algorithm.

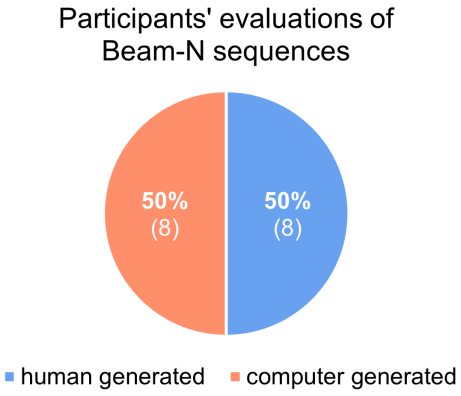


Figure 4.18: Distribution of the participants' evaluations when presented with a sequence sampled with Beam-N.

An unexpected 50% of participants correctly detected that the sequences generated by our model were computer generated. Such a high percentage clearly indicates that sequences sampled with Beam-N are not similar to the real sequences. As mentioned earlier, during these experiments the participants provided some feedback which led us to develop the variation Beam-3. The results for this sampling algorithm are shown in Figure 4.19.

Restricting predicted sequences to a maximum length of 3 clearly had a positive influence on the performance of our model. The success rate at "tricking" participants increased from 50% to 88%, even higher than the percentage of real sequences that were classified as "human generated" by the participants. One possible explanation for the fact that these sequences were classified as more human-like than actual human sequences is that our model learned more general principles for the task that the

majority of humans adhere to, instead of learning specific principles that only a few humans consider, thus pleasing more participants. This is supported by the fact that the model derives from data gathered from many different participants, thus improving its variability and reducing the probability of overfitting.

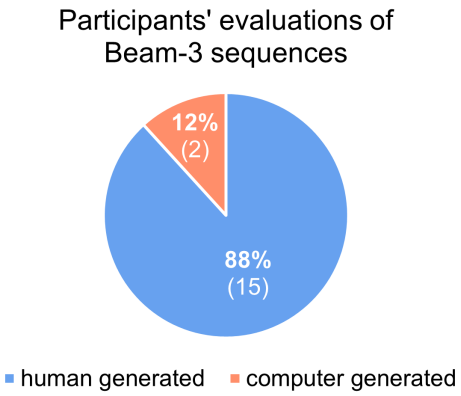


Figure 4.19: Distribution of the participants' evaluations when presented with a sequence sampled with Beam-3.

All these experimental results are summarized in Table 4.2. It is important to note that since the Beam-3 sampling strategy was introduced already during the course of the experiment, both beam search sampling strategies have about half the samples of the other strategies.

Table 4.2: Results from the sequence prediction experiment.

	Participants evaluations		
Sequence Type	*Human generated*	*Computer generated*	Number of samples
Random	7%	**93%**	30
Real	**79%**	21%	33
Beam-N	50%	50%	16
Beam-3	**88%**	12%	17

Although this intuitive analysis of the results provides a detailed understanding of the model's performance, we also conduct a statistical analysis to validate the significance of the data. As addressed in Section 3.5, the 2-proportions Z-test can be applied to verify if there is a statistical difference between the proportion of a parameter across two populations. However, its derivation from the central limit theorem demands a sufficient sample size for it to be applicable. In our results, both Beam sampling strategies have fewer than 30 samples and only two samples for the evaluation "computer generated" for the Beam-3 strategy. These limited sample sizes render the 2-proportions test inapplicable.

Instead, we conduct a Boschloo's exact test [46] to test whether the proportion of individuals that was "tricked" by the Beam-3 algorithm, p_1, is greater than the proportion of individuals "tricked" by Beam-N,

p_2, at a significance level of $\alpha = 0.05$. As such, our null and one-tailed alternative hypotheses are

$$H_0 : p_1 = p_2 \tag{4.1}$$

$$H_1 : p_1 > p_2 \, . \tag{4.2}$$

From this test we obtain a p-value of 0.01, and hence, at this significance level, we reject the null hypothesis and prove the difference between these two proportions. In other words, Beam-3 is an improvement over Beam-N. However, this only guarantees that there is an effect, and *not* that it is significant. Recall from Section 3.5 that the effect size measures importance or significance on a normalized scale, regardless of the quantities in the study.

There are multiple measures of this effect size, but we consider Cohen's H [48], as is standard in statistical analysis. Our results produce an h value of approximately 0.86, which indicates a large effect according to Cohen's interpretation [48]. Finally, these results are validated with a post-hoc power analysis, reporting a power of 79%, which is a satisfactory power value according to [48].

5

Conclusions

Contents

5.1 Conclusions

This thesis addressed the task of packing everyday objects. We proposed a model to predict the packing sequence for a set of objects, derived from the new dataset of bin packing demonstrations that we collected and made publicly available.

Unlike most previous research, our goal was to learn some aspects of this task directly from human experts instead of relying on hand-crafted heuristics or reinforcement learning. In particular, we learned to predict an optimal packing sequence for a random subset of objects from expert demonstrations. The proposed approach has the advantage of requiring a fast and simple training scheme. Furthermore, we showed how the resulting model can be sampled with a modified version of beam search to predict packing sequences in real time.

Our experimental results proved that our model surpassed human performance at generating packing sequence predictions that individuals classified as human-like. More importantly, this high success rate at predicting human-like packing sequences revealed that the proposed approach was successful at extracting implicit task knowledge directly from human demonstrations. This highlights that learning from human demonstrations is a noteworthy approach to the bin packing task and has the potential to lead to superior models. This is especially important in the context of collaborative robots where producing legible actions is a top priority.

This model is an example of how our dataset, BoxED, can be leveraged to learn methods that are as effective at the bin packing task as humans. This dataset is the first public collection of human demonstrations for this task and is very diverse in the number of parameters that are recorded. Furthermore, by collecting these data in virtual reality we were able to accurately record task parameters such as object poses, which would otherwise be very difficult.

Moreover, this dataset also gathered data on multiple aspects that are not exclusive to the bin packing task. For instance, the grasping data can be used for robotic grasp inference and the object trajectories can be used to better understand and model how humans manipulate objects. Since only a portion of these data was used in training our model, we argue that the dataset still has a significant amount of unexploited potential.

5.2 Future Work

Future work may include testing other sampling mechanisms to predict packing sequences from our Markov chain model. These could include comparing our current Beam-3 sampling method to greedy or epsilon greedy search. To address the limitation of depending explicitly on the name of each object, which limits the maximum number of objects that our Markov chain can tolerate, we could also introduce

a level of abstraction from the objects. One possibility for this improvement would be to use some of the objects' physical properties instead of their names. For instance, the objects could be classified into different levels of fragility and characterized according to their shape. These parameters could be estimated from an RGB-D image of the object and used to learn a new Markov chain.

Furthermore, the data we collected has the potential to create the first supervised learning approach to predicting a placement pose for irregular objects (such as groceries) in the bin packing task. This would likely involve the end-to-end training of a neural network to predict a placement pose for an object given the object and the current layout of the container. If this neural network were to receive images as input then higher-quality images could be created in a more realistic simulator, such as Blender, by recreating the demonstrations from the data that we collected. This would also be the first approach in this field to predict placement poses from human demonstrations.

Finally, the culmination of this work would be to develop and deploy a complete system for packing irregular objects on a real robot.

Bibliography

[1] Ocado Group, (accessed Aug. 8, 2022). [Online]. Available: https://www.ocadogroup.com/

[2] Amazon Inc., "Introducing Amazon Astro, household robot for home monitoring," (accessed Jan. 10, 2022). [Online]. Available: https://www.amazon.com/Introducing-Amazon-Astro/dp/B078NSDFSB

[3] temi USA inc., "temi - The personal robot," (accessed Jan. 10, 2022). [Online]. Available: https://www.robotemi.com/

[4] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2013, pp. 301–308.

[5] J. E. Colgate and M. A. Peshkin, "Cobots," U.S. Patent 5 952 796, Sep. 14, 1999.

[6] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A Survey on Learning-Based Robotic Grasping," *Current Robotics Reports*, 2020.

[7] Z. Long, Q. Jiang, T. Shuai, F. Wen, and C. Liang, "A Systematic Review and Meta-analysis of Robotic Gripper," *IOP Conference Series: Materials Science and Engineering*, vol. 782, no. 4, mar 2020.

[8] J. Choi, "Automatic, Careful Online Packing of Groceries Using a Soft Robotic Manipulator and Multimodal Sensing," M.S. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2022.

[9] R. Hoggett. 1948 - GE Master-Slave Manipulator. Accessed on: Dec. 28, 2021. [Online]. Available: http://cyberneticzoo.com/teleoperators/1948-ge-master-slave-manipulator-john-payne-american/

[10] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 03 1991.

[11] A. Y. Ng and S. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[12] T. Munzer, M. Toussaint, and M. Lopes, “Efficient Behavior Learning in Human-Robot Collaboration,” *Autonomous Robots*, vol. 42, no. 5, p. 1103–1115, June 2018.

[13] S. Dasari and A. K. Gupta, “Transformers for One-Shot Visual Imitation,” in *Conference on Robot Learning (CoRL)*, Nov. 2020.

[14] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, “Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-to-End Learning from Demonstration,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3758–3765.

[15] A. Ghadirzadeh, X. Chen, W. Yin, Z. Yi, M. Björkman, and D. Kragic, “Human-Centered Collaborative Robots With Deep Reinforcement Learning,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 566–571, Apr. 2021.

[16] P. Wang, F. Manhardt, L. Minciullo, L. Garattoni, S. Meier, N. Navab, and B. Busam, “DemoGrasp: Few-Shot Learning for Robotic Grasping with Human Demonstration,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 5733–5740.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[18] F. Esposito, C. Pek, M. C. Welle, and D. Kragic, “Learning Task Constraints in Visual-Action Planning from Demonstrations,” in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2021, pp. 131–138.

[19] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-End Driving Via Conditional Imitation Learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4693–4700.

[20] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB ’94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 487–499.

[21] M. J. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences,” in *Machine Learning*, 2001, pp. 31–60.

[22] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[23] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock Price Prediction Using LSTM, RNN and CNN-sliding Window Model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647.

[24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, vol. 9, 1997, pp. 1735–1780.

[25] S. Ali, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira, "On-line three-dimensional packing problems: A review of off-line and on-line solution approaches," *Computers and Industrial Engineering*, vol. 168, p. 108122, 2022.

[26] F. Wang and K. Hauser, "Robot Packing With Known Items and Nondeterministic Arrival Order," *IEEE Transactions on Automation Science and Engineering (T-ASE)*, vol. 18, no. 4, pp. 1901–1915, 2021.

[27] L. Duan, H. Hu, Y. Qian, Y. Gong, X. Zhang, J. Wei, and Y. Xu, "A Multi-task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem," in *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2019, pp. 1386–1394.

[28] 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, nov 2020.

[29] Y. Jiang, Z. Cao, and J. Zhang, "Solving 3D Bin Packing Problem via Multimodal Deep Reinforcement Learning," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2021, p. 1548–1550.

[30] I. Ikonen, W. E. Biles, A. Kumar, J. C. Wissel, and R. K. Ragade, "A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects Having Cavities and Holes," in *Proceedings of the 7th International Conference on Genetic Algorithms*, T. Bäck, Ed. Morgan Kaufmann, 1997, pp. 591–598.

[31] Y. Ma, Z. Chen, W. Hu, and W. Wang, "Packing Irregular Objects in 3D Space via Hybrid Optimization," *Computer Graphics Forum*, vol. 37, 2018.

[32] T. Romanova, J. Bennell, Y. Stoyan, and A. Pankratov, "Packing of Concave Polyhedra with Continuous Rotations Using Nonlinear Optimisation," *European Journal of Operational Research*, vol. 268, no. 1, pp. 37–53, 2018.

[33] F. Wang and K. Hauser, “Stable Bin Packing of Non-convex 3D Objects with a Robot Manipulator,” in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8698–8704.

[34] H. Zhao, Y. Yu, and K. Xu, “Learning Efficient Online 3D Bin Packing on Packing Configuration Trees,” in *Proceedings of the 2022 International Conference on Learning Representations (ICLR)*.

[35] R. Verma, A. Singhal, H. Khadilkar, A. Basumatary, S. Nayak, H. V. Singh, S. Kumar, and R. Sinha, “A Generalized Reinforcement Learning Algorithm for Online 3D Bin-Packing,” *Computing Research Repository (CoRR)*, vol. abs/2007.00463, 2020.

[36] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, “Online 3D Bin Packing with Constrained Deep Reinforcement Learning,” in *35th AAAI Conference on Artificial Intelligence1*. AAAI Press, 2021, pp. 741–749.

[37] Y. Aquilina and M. A. Saliba, “An Automated Supermarket Checkout System Utilizing a SCARA Robot: Preliminary Prototype Development,” *Procedia Manufacturing*, vol. 38, pp. 1558–1565, 2019.

[38] S. Song, A. Zeng, J. Lee, and T. Funkhouser, “Grasping in the Wild: Learning 6DoF Closed-Loop Grasping From Low-Cost Demonstrations,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, pp. 4978–4985, 2020.

[39] Unity Technologies. Unity. Accessed on: May, 12 2022. [Online]. Available: https://unity.com/

[40] NVIDIA. Nvidia physx. Accessed on: Sep, 22 2022. [Online]. Available: https://developer.nvidia.com/physx-sdk

[41] N. Tax, I. Teinemaa, and S. J. van Zelst, “An Interdisciplinary Comparison of Sequence Modeling Methods for Next-Element Prediction,” *Software and Systems Modeling (SoSyM)*, vol. 19, no. 6, p. 1345–1365, nov 2020. [Online]. Available: https://doi.org/10.1007/s10270-020-00789-3

[42] M. Kumar and P. Srinivas, “Algorithms for Mining Sequential Patterns,” *International Journal of Information Sciences and Application*, vol. 3, no. 1, pp. 59–69, 2011.

[43] A. N. Langville and P. von Hilgers, “The Five Greatest Applications of Markov Chains,” 2006.

[44] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[45] R. A. Fisher, “On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P,” *Journal of the Royal Statistical Society*, vol. 85, no. 1, pp. 87–94, 1922.

[46] R. D. Boschloo, “Raised conditional level of significance for the 2 $\times$ 2-table when testing the equality of two probabilities,” *Statistica Neerlandica*, vol. 24, no. 1, pp. 1–9, 1970.

[47] D. V. Mehrotra, I. S. F. Chan, and R. L. Berger, “A Cautionary Note on Exact Unconditional Inference for a Difference between Two Independent Binomial Proportions,” *Biometrics. Journal of the International Biometric Society*, vol. 59, no. 2, pp. 441–450, 2003.

[48] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Routledge, 1988.

[49] M. E. Bartlett, C. E. R. Edmunds, T. Belpaeme, and S. Thill, “Have I Got the Power? Analysing and Reporting Statistical Power in HRI,” *ACM Transactions on Human-Robot Interaction*, vol. 11, no. 2, pp. 1–16, Jun. 2022.

[50] U. of Dusseldorf. G*Power: Statistical Power Analyzes for Mac and Windows. Accessed on: Sep, 21 2022. [Online]. Available: https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower

[51] 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 Robotics Automation Magazine (RAM)*, vol. 22, no. 3, pp. 36–52, 2015.

[52] D. Crockford. Introducing JSON. Accessed on: May, 10 2022. [Online]. Available: https://www.json.org/

[53] N. Jakobi, P. Husbands, and I. Harvey, “Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics,” in *Advances in Artificial Life*, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[54] The Blender Foundation. Blender - a 3D Modeling and Rendering Package. Accessed on: Oct., 7 2022. [Online]. Available: https://blender.org/

[55] A. Hagberg. A Python Interface to the Graphviz Graph Layout and Visualization Package. Accessed on: Oct, 7 2022. [Online]. Available: https://pygraphviz.github.io/