MANIPULATION AND PLACEMENT PLANNING FOR LOADING A DISHWASHER BY A ROBOT

by

Bahar İrfan

B.S., Mechanical Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2016

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my thesis supervisor, Prof. H. Levent Akın, for his advice throughout my thesis.

I would like to thank Assist. Prof. Albert Ali Salah for all of his advice on designing the social questionnaires and the observational study, which enabled me to turn this work into a social robotics project. I am deeply grateful to him and to Prof. H. Işıl Bozma for kindly accepting to be the members of my thesis committee, and also for their patience.

I would also like to thank Assist. Prof. Emre Uğur, Assist. Prof. Evren Samur, Prof. Mahmut Ekşioğlu, Assoc. Prof. Caner Taşkın and Assist. Prof. Mustafa Gökçe Baydoğan for giving me advice on different parts of my project.

I am indebted to Özgür Erkent and his group, who provided and modified their object recognition algorithm for using it in this project, and also to N. Ergin Özkucur for helping me connect the arm to the *ROS MoveIt!* system.

I am grateful to the members of Robot Lab for helping me during this project, and for letting me use most of the tables in the laboratory for my experiments. Moreover, I would like to thank all of the participants of the social studies conducted for this thesis, for their time and efforts.

I am deeply thankful to my friends who have supported and helped me throughout my thesis. I would especially like to thank K. Yasemin Usta, N. Cihan Camgöz and Serhan Daniş for all their invaluable help and support.

Last but not least, I would like to thank my mother and my father, Hanife and N. Coşkun İrfan, my sister E. Pınar İrfan, my grandmother Yüksel Carban, my uncle Haluk Carban, and all the other members of my family for supporting me and giving me advice on my project and for everything else. I am very lucky to have them in my life.

I would like to dedicate this thesis to the deceased members of my family, to both of my grandfathers İsmail İrfan and Salih Carban, my grandmother Şekibe İrfan, and my grand-aunt Leman Büyükçubukçu. I wish they were still here with me on this day.

ABSTRACT

MANIPULATION AND PLACEMENT PLANNING FOR LOADING A DISHWASHER BY A ROBOT

Placing the dishes into a dishwasher can take a considerable amount of time, which is a process a person would have to repeat every few days. Therefore, we decided to build a robotic system which would place the mugs on the counter-top to the dishwasher tray. Three types of problems are solved in order to realize this task: object recognition for detecting the mugs; 2D packing problem for placement planning of mugs on the dishwasher tray; placing the mugs into the dishwasher through manipulation. These tasks are solved within a simulation environment and in the physical world by a 5 degree-of-freedom robot arm and a depth camera. Our contribution with this project is placement planning, in which we use different heuristics and optimization methods with different cost methods and functions. Top-Left-Fill (TLF) and Left-Top-Fill (LTF) heuristics are used for packing and Simulated Annealing (SA), Genetic Algorithm (GA), and two different Particle Swarm Optimization (PSO) methods are used for finding the near-optimal solution for the placement of the mugs. The initial configurations of our methods are made with closest-item-first, sorted-size heuristics and random order. In our optimization algorithms, we use weighted-sum (WS) and ranking fitness (RF) methods with bin-packing cost (BPC), user preference-based cost (UPC) and engineering cost (EC) functions. In order to find the parameters and their weights for the cost functions, we conducted preliminary and user studies. Furthermore, we made an observational study to examine the placement methods of the participants. An annotation system was made to rate the sortedness of these placements with a Likert scale. The sortedness of the placements were compared with the Obsessive-Compulsive Scale test results from the observational study to observe the relation. The results of our simulation experiments show that, in general, the GA with the LTF heuristic produces the lowest costs with the WS method, whereas for the RF method, the best optimization method is the SA.

ÖZET

BULAŞIK MAKİNESİNİ ROBOTLA DOLDURMAK İÇİN MANİPULASYON VE YERLEŞTİRME PLANLAMASI

Bulaşıkları bulaşık makinesine yerleştirmek zaman alır ve birkaç günde bir tekrar edilmesi gerekir. Bu yüzden, bardakları makineye yerleştiren robotik bir sistem yapmaya karar verdik. Bunu gerçekleştirebilmek için üç farklı sorunun çözülmesi gerekti: tezgahın üstündeki bardakları tanıma; 2 boyutlu yerleşim planlama; robot kolla yerleştirmek için hareket planlaması. Bu sorunların hepsinin çözümleri simülasyon ortamında ve gerçek dünyada 5 serbestlik derecesine sahip bir robot kol ve derinlik kamerasıyla yapıldı. Bizim bu projeyle katkımız farklı buluşsal yöntemleri ve optimizasyon metodlarını maliyet fonksiyonları ve metodlarıyla kullanarak yerleştirme planlaması yapmamızdır. Üst-Sol-Doldurma ve Sol-Üst-Doldurma yöntemlerini nesneleri paketlemek için ve Benzetilmiş Tavlama, Genetik Algoritma ve iki farklı Parçacık Sürü Optimizasyonu metodlarını optimuma yakın bir sonuç bulmak için kullanıyoruz. Metodlarımızdaki ilk yerleşim düzeni, en yakın nesneyi önce yerleştirme, boyutlarına göre veya rastgele yerleştirme yöntemlerinden biri ile bulunuyor. Ağırlıklı-toplam ve uyumları sıralama yöntemleri, yerleştirme problemi maliyeti, kullanıcı tercihine bağlı maliyet ve mühendislik maliyeti fonksiyonlarıyla beraber optimizasyon algoritmalarımızda kullanılıyor. Fonksiyonlardaki parametreleri ve ağırlıklarını bulmak için ön araştırma ve kullanıcı araştırmaları yapıldı. Bunun üzerine, insanların yerleştirme biçimlerini gözlemlemek için gözlemsel araştırma yapıldı. İnsanların yerleştirmelerini düzenlilik kriterine göre Likert ölçeğiyle puanlayabilmek için açımlama sistemi yapıldı. Buradan elde edilen değerler obsesif kompulsif bozukluğun düzenlilik kriteriyle ilişkisini bulabilmek için gözlemsel araştırmadaki değerlerle karşılaştırıldı. Simülasyon sonuçlarımız, genelde, ağırlıklı-toplam metodu için Sol-Üst-Doldurma yöntemiyle kullanılan Genetik Algoritmanın en az maliyeti doğurduğunu gösterdi. Uyumları sıralama yöntemiyle ise Sol-Üst-Doldurmayla kullanılan Benzetilmiş Tavlama yönteminin en az maliyeti olduğu gözlemlendi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Arm base
a_{DH}	Denavit-Hartenberg parameter for the distance between Z axes of the coordinate frames
A	Covered area by the mug(s) within the tray
A'	Normalized covered area
$\vec{B}_l$	Lower bound vector for the minimum position
$\vec{B}_u$	Upper bound vector for the maximum position
c	Counter-top
c_i	Acceleration coefficient in Particle Swarm Optimization, where $i = 1,2,3$
C	Cost function
$\vec{C}$	Vector of cost function parameters $((1-n'),t',E',(1-s))$
C_b	Bin-packing cost function
C_u	User preference-based cost function
C_e	Engineering cost function
cl	Cell
cr	Cooling rate in Simulated Annealing
d_{DH}	Denavit-Hartenberg parameter for the distance between the origin O_{i-1} and the intersection of the X_i axis with Z_{i-1}
ds	Dataset number
e	Energy of the placement configuration in Simulated Annealing
E	Electrical energy
E'	Normalized electrical energy
f_m	Fitness method
f_f	Fitness function
F	Force exerted by the arm on the mug
g	Gripper
G	Gravitational acceleration
h	Packing heuristic (Top-Left-Fill) or (Left-Top-Fill)

l	Length
lbg	Parameter for checking if local best is chosen as the global best
m	Mug
M_m	Mass of a mug
$min(w_m)$	Minimum width of the mugs in the dataset
$min(w_r)$	Minimum radius of the mugs in the dataset
n	Number of placed mugs
n_ρ	Placed value of the particle for Particle Swarm Optimization with position (1 if the particle is placed on the tray, 0 otherwise)
n'	Normalized number of placed mugs
N	Number of cost function parameters
o	The order of placed items
O	The objective function value in Particle Swarm Optimization with position
os	Offspring chromosome in Genetic Algorithm
om	Occupancy matrix for packing
p	Particle in Particle Swarm Optimization
$P(e, e', T)$	Acceptance probability function in Simulated Annealing
pso_m	Particle Swarm Optimization method
PS	Probability of choosing improved Particle Swarm Optimization
r	Radius
R_{ij}	Number of responses in the user study for a parameter i with rank j
s	Sortedness
S	Configuration in Simulated Annealing
t	Time
t'	Normalized time
t_c	Placement calculation time
T	Temperature in Simulated Annealing
T_{abs}	Absolute temperature in Simulated Annealing
tr	Tray
$\vec{v}$	Velocity vector for a particle

V_a	Speed of the arm
w	Width
W_i	Weight of parameter i in the cost function
x	x coordinate
$\vec{x}$	Position vector in Particle Swarm Optimization
x_{Lg}	Lower x limit for the gripper
x_{Ltr}	Lower x limit for the tray
x_{Ug}	Gripper upper x limit
x_{Utr}	Upper x limit for the tray
X	Axes in the forward direction of the robot
y	y coordinate
y_{Ltr}	Lower y limit for the tray
y_{Utr}	Upper y limit for the tray
Y	Axes in the horizontal (left) direction to the robot
z	z coordinate
Z	Axes in the upwards direction to the robot
α	Angle of the target position of the gripper with respect to the global world frame
α_{DH}	Denavit-Hartenberg parameter for the angle between Z axes
α_{Lx}	Target gripper angle for lower x limit
α_{Ux}	Target gripper angle for upper x limit
α'	Angle of the target position of the gripper with respect to the arm base
β'	Mug angle with respect to the arm base
γ'	Angle between the target position of the gripper and the mug center with respect to the arm base
Δm	Euclidean distance traversed to place a mug
Δ_{ma}	Mug center distance to arm base
Δ_{mg}	Mug center distance to gripper
ζ_o	Overlap function in Particle Swarm Optimization with position
ζ_r	Is in region function in Particle Swarm Optimization with position

Θ	Key of a particle in Particle Swarm Optimization with key
θ'	Angle between gripper and mug center and the length of the gripper with respect to the arm base
θ_{DH}	Denavit-Hartenberg parameter for the angle between X axes
ι	Item number in the original packing order
κ	Sort method (initial placement order of the mugs)
λ	Chromosome in Genetic Algorithm
λ_b	Best chromosome in a generation in Genetic Algorithm
Λ	Placement order set of the mugs
π	Mathematical constant
ρ	Particle in Particle Swarm Optimization
ρ_{gb}	Global best particle in Particle Swarm Optimization
ρ_{g2b}	Global second best particle in Particle Swarm Optimization
ρ_{lb}	Local best particle in Particle Swarm Optimization
$\hat{\rho}_i$	Mutated particle i in Particle Swarm Optimization with position
ρ	Population in Genetic Algorithm
ς	Gene in Genetic Algorithm
τ	Iteration number in an optimization method
τ_{max}	Maximum number of iterations allowed in an optimization method
ϕ	Vector of placement parameters (A, t_c, t, E, s)
χ	Particle ordering in Particle Swarm Optimization
χ_b	The best particle ordering that has the lowest cost for the method
Ψ	Fitness function parameters: $(n, C, \vec{C}, o, \phi)$
ω	Inertia in Particle Swarm Optimization
ω_{max}	Maximum inertia in Particle Swarm Optimization
ω_{min}	Minimum inertia in Particle Swarm Optimization

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
BF	Best-Fit heuristic
BH	Basic heuristic methods (CIS, NIS, NDS, RO)
BL	Bottom-Left heuristic
BLD	Bottom-Left-Decreasing
BLF	Bottom-Left-Fill heuristic
BPC	Bin-packing cost
BRF	Bottom-Right-Fill heuristic
CAD	Computer-aided design
CIF	Closest-item-first heuristic
DH	Denavit-Hartenberg parameters
DOF	Degree of Freedom
EC	Engineering cost
FSR	Force-Sensing Resistor
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
ICP	Iterative Closest Point
KDL	Kinematics and Dynamics Library
LBF	Left-Bottom-Fill heuristic
LTF	Left-Top-Fill heuristic
NDS	Non-decreasing size
NE	Naïve Evolution
NIS	Non-increasing size
NN	Nearest Neighbor
OCD	Obsessive-Compulsive Disease
PID	Proportional–integral–derivative controller
PSO	Particle Swarm Optimization

PSOp	Particle Swarm Optimization with mug positions as particles
PSOp$_i$	Improved Particle Swarm Optimization with mug positions as particles
PSOp$_n$	Normal Particle Swarm Optimization with mug positions as particles
PSOk	Particle Swarm Optimization with keys as particles
PSOk$_i$	Improved Particle Swarm Optimization with keys as particles
PSOk$_n$	Normal Particle Swarm Optimization with keys as particles
RANSAC	Random Sample Consensus
RBF	Right-Bottom-Fill heuristic
RF	Ranking fitness method
RGB-D	Red Green Blue-Depth, depth image format
RLPA	Rectangular layer-packing algorithm
RMSE	Root Mean Square Error
RO	Random order configuration
RS	Random search method
RTF	Right-Top-Fill heuristic
SA	Simulated Annealing
SRDF	Semantic Robot Description Format
STL	Stereolithography
TLF	Top-Left-Fill heuristic
TRF	Top-Right-Fill heuristic
TS	Tabu Search algorithm
UPC	User preference-based cost
URDF	Unified Robot Description Format
WS	Weighted-sum fitness method

1. INTRODUCTION

Service robotics is an emerging field which aims to increase the use of robots in our daily lives with potential applications such as helping patients or elderly, and doing household chores. We waste a considerable amount of time and energy in our daily lives to do housework and we repeat the same process at least every two-three days. Whether it is cleaning up the dishes after a meal or doing laundry, this wasted time can be used more efficiently. Hence, many studies have been conducted in this area. From pool cleaners such as Dolphin robots [1] to vacuum cleaner robots such as iRobot Roomba [2], many robots are now affordable to be used domestically. Currently, there are no robots available for purchase in the market that can do other house chores such as washing the dishes or loading the washing machine.

In case of robotic dishwashers, one of the problems for loading a dishwasher is grasping the object without breaking it. Many studies have been conducted in that area, and the most common approach is to detect the object at hand and use a previously determined or learned force depending on the object [3,4]. Learning online or from a person have also been studied, which even though increases the risk of breaking the object, makes the process more flexible [5, 6].

In University of Tokyo, JSK Laboratory, a humanoid robot, HRP2-JSK, was used to do household chores, such as washing dishes, sweeping the floor, vacuuming and pouring water from a teapot [7]. The robot can wash the dishes with its manipulators under the water using waterproof gloves. They use vision for verifying the state of the action such as turning on the water outlet, and detecting the position of a dish when picking or placing it. They also used behavior verification for checking if the current behavior was successful in providing the desired outcome. The steps of the robot were pushing the tap, detecting the water flow, detecting humans for safety, washing a dish, and placing a clean dish next to the sink. Panasonic has produced a robot arm in 2008, in cooperation with Information and Robot Technology (IRT) Laboratory of University of Tokyo, which would rinse dishes and then place them into the dishwasher [8]. They used grasp and manipulation planning,

and collision detection algorithms, however, the product has never become available for purchase [9]. The system had 8 DOF and 22 sensors that resulted in an expensive arm design and higher computational cost due to the high number of sensors since values from each sensor should be checked for manipulation planning. Stanford University's STAIR robot can unload the dishwasher with the ability to grasp novel objects using vision [4]. They used a learning algorithm for detecting grasp points using depth-based features through a stereo camera, however, loading the dishwasher was done by a human loader. Karlsruhe Institute of Technology Humanoids and Intelligence Systems Lab has conducted studies on grasp and motion planning and their robot ARMAR-III can grasp and place objects into a dishwasher, but it places them to the available spaces without optimization of the overall placement [10]. Cornell University has studied the placement of objects in a scene [11]. They used human preferences about placements and the geometric relationship between objects and their placing environments using the max-margin learning approach that incorporates stability of the placed object, semantic preference, and stacking criteria. They placed plates and different types of objects into an actual tray of the dishwasher, however, did not use a placement algorithm within the tray. In addition, their algorithm places mugs and glasses upright in the tray, which would be inefficient for the cleaning process. In the University of Birmingham, they have built the Boris robot as a part of Probabilistic and Compositional Representations of Object Manipulation (PaCMan) project in order to clean up the table, load and unload the dishwasher. The main concern in their study is object manipulation, which they have realized with dexterous robotic hands [3, 12]. They also used a box instead of the actual dishwasher tray, but in their demonstrations of their project, the object is dropped from a relatively high distance to the box, which is not preferable since normal dishes can be broken when they are dropped at that height. In Soongsil University, they used a bin-packing method, the branch-and-bound algorithm with Finite First-Fit, Finite Best-Fit and modified Finite Best-Fit and later proposed their own methods, for the optimization of a dish cart storage loaded by a service robot [13, 14]. They used several types of dishes such as plates, cups, forks, and knifes, but they did not use a robot for the manipulation or the object detection, and they only simulated the container loading results.

In this study, we aim to develop a robot, which can take mugs from the kitchen counter and place them into the dishwasher. We aim to solve a common problem in households,

which is to place the dishes that can pile up on the kitchen counter after a busy week. Therefore, we want to provide a system solution which would be able to place items, without the need for human intervention in the process. We aim to make a full low-cost robot system consisting of a robot arm and a depth camera, which can detect mugs through the camera, and perform manipulation and placement planning. Our goal is maximizing the number of mugs placed into the dishwasher, by using a placement optimization method, for decreasing the number of times the dishwasher needs to be run so that water and energy consumption would be decreased. The other criteria within the study are minimizing the energy for manipulation and minimizing the time for calculation and manipulation. The energy needs to be minimized in order to decrease the electricity or battery consumption by the motors of the robot, which requires reducing the total distance traveled by the robot arm. The time for calculation and manipulation should be minimized, because the whole process will be done in a household, therefore it will not be preferable for the robot to take a long time for this process. As a result, efficient algorithms are required for detection, manipulation and placement planning, and a minimum number of moves should be made the robot arm in order to decrease the time for manipulation. In order to optimize these parameters, and find out whether any other parameter exists for optimization, we decided to determine the cost function which would affect the placement of the robot, according to the people's preferences. Hence, we conducted a preliminary study to find the parameters of our cost function. Afterward, using those parameters that are relevant to our study and can be optimized, we conducted a user study for finding the weights of those parameters within our cost function.

The difference between all the related studies in placing dishes by the use of a real robot platform and our study is that in these projects, the placement planning within the dishwasher for placing the maximum number of dishes is not the concern, hence, the placement may not be done efficiently. But in our project, we introduce a system, in which we study placement, grasp, and manipulation planning. We also add the people's preferences by using cost functions in our placement. We use a force-sensing resistor on the inner surface of the gripper and a previously determined force for grasping objects. In our project, we release the mugs slightly above the surface of the tray, so that they will not have a risk of breaking.

We make some assumptions in order to decrease the complexity of the placement plan-

ning algorithms. The placement problem can be viewed as a 2-dimensional (2D) bin packing problem since the mugs cannot be placed on top of each other on a tray. The tray of the dishwasher is modeled as an empty box. The mugs are assumed to be in an upright position. We also assume that there are only mugs on the counter-top, that is, we were not concerned with detecting and avoiding humans, other dishes or other objects, in our study.

The organization of the rest of the thesis is as follows. The related background for the bin packing problems and the state of art in 3D object recognition are presented in Chapter 2. The system design and the system requirements are presented in Chapter 3. The methodology of our approach is presented in Chapter 4. Firstly, the object recognition method used for detecting mugs on the counter-top is described. Afterward, the placement planning methods by using the optimization planning algorithms Simulated Annealing, Particle Swarm Optimization and Genetic Algorithm and our proposed approaches to these algorithms are presented. Furthermore, the planning method for placing mugs on the counter-top to the tray are explained, along with the use of a force sensing resistor for recognizing the grasps. Finally, in this chapter, we describe the use of the cost functions to be used as the fitness functions for our optimization algorithms. The experimental results for object recognition, and the simulation and real robot test results for each of the methods described in the methodology are presented in Chapter 5. In this chapter, we also describe our preliminary and user studies and the annotation system, which were used as baselines for creating the cost functions used in the placement tests. We discuss our methods and conclude the dissertation with the future directives for improving our system in Chapter 6.

2. RELATED WORK

Our study aims to create a system that can autonomously detect mugs on a countertop, manipulate them and optimize their placement within the tray according to some criteria. Therefore, it is required to combine object recognition and optimization in packing and placement planning, hence, we conducted a thorough literature survey on the available methods in these fields.

2.1. Bin Packing Problem

In many industrial applications, it is required to allocate a set of rectangular items to larger rectangular stock units by minimizing the waste. In these applications, the common objective function is to pack all the requested items into minimum number of units. These optimization problems are known in the literature as two-dimensional bin packing problems, with given width and height of the bins [15].

The packing problems are categorized based on the characteristics of each problem. The characteristics and the elementary types of the bin packing problems are given below [16].

- **Dimensionality:** The minimum number of dimensions required to describe the geometry of the patterns. Elementary types are 1-dimensional, 2-dimensional, 3-dimensional, and n-dimensional problems (more than three dimensions)
- **Quantity measurement:**
  (i) Discrete (or "integer") measurement: This refers to frequency or number of objects or items of a certain shape
  (ii) Continuous (or "fractional") measurement, by real numbers: This measures the whole length of several objects or items having the same shape with respect to the relevant dimensions, the length of the objects or items with respect to another dimension are not essential for the geometry of the patterns
- **Shape of figures:** The form, size, and orientation uniquely determines a figure. For

multi-dimensional problems, there are two forms of the figure: regular and irregular. Depending on the dimensionality, the length, area, or volume of a figure may be used as its size. Examples for 2D regular forms are circles, squares, and rectangles with distinct ratios of width and length. For irregular objects:

(i) If the orientation is allowed then similar objects and items may not be distinguished. If only 90-degree turns are allowed then the objects and items are considered as identical.

(ii) If the orientation is fixed then similar objects and items are differentiated except those which can be made identical by translations.

- **Availability:** This refers to the lower and upper bounds of the quantity, the order and the sequence of the items, and the date when an object or item can be or has to be cut or packed. Classical bin packing and cutting stock problems are generally concerned with an infinite number of objects. Bin packing considers only a few items for each small figure in which there are many different figures, on the other hand, cutting stock problems delays with a lot of items for each small figure and the figures are similar with only a few different ones.
- **Pattern restrictions:** There are four groups of pattern restrictions.
  (i) The minimum or the maximum distances between small figures or between cuts for dividing large figures are important, e.g. when breaking glass or loading containers.
  (ii) The relative orientation of the small figures to each other or to the large figure are important, e.g. when cutting patterned fabrics or when loading fragile goods on pallets.
  (iii) The frequency of small items or figures in a pattern, specifically regarding the combination of a number of different small figures or items, e.g. maximum amounts of waste and the maximum number of ranges of sizes or orders.
  (iv) Type and number of permitted cuts are important restrictions, especially if the objects and items are of rectangular or of block form.
- **Objectives:** The criterion to be maximized or minimized. The criteria can be the constraints of the cutting or packing process and it can be a single objective or combination of the objectives that refer to the optimization of:

(i) Quantities of large or small objects which make up the patterns

(ii) Geometry of the patterns (layout-optimization)

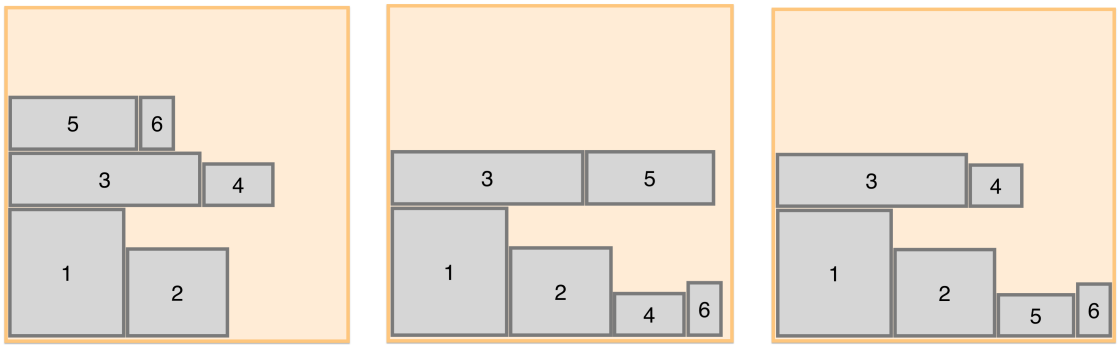
(iii) Sequence, combination, or number of patterns

2.1.1. Packing Strategies and Heuristics

There are two types of algorithms depending on the availability of the items to be packed beforehand: offline algorithms, which have full knowledge of the inputs, and online algorithms, which packs the items in the order they encountered in the scan of the input, that is, without knowledge of the next input. There are two basic strategies for heuristics: the divide-and-conquer method and the iterative improvement method. Divide-and-conquer methods divide the problem into smaller subproblems and then try to solve these subproblems. The solutions to the subproblems must then be brought together in a way to solve the full problem. In iterative improvement, an initial known configuration is available and a standard rearrangement operation is applied until a configuration that improves the cost function is found. The calculated rearranged configuration becomes the new configuration of the system, and the process is carried on until no further improvements can be found. The iterative improvement consists of a search in this coordinate space for the rearrangement steps which lead downhill. This search can get stuck in a local optimum, hence the process is carried out many times, by starting from different randomly generated configurations, and the best result is saved. Permutation coding scheme is used in which a permutation is constructed and then the items are placed one by one according to the permutation. For the first phase, a standard strategy to construct a permutation is by giving a larger item a higher priority than a smaller one. In a level algorithm, the placement is obtained by placing the items from left to right in rows forming levels. The first level is the bottom of the object, and each subsequent level is along the horizontal line coinciding with the top of the tallest item packed on the level below.

The most popular level algorithms are the next fit, first fit and best fit strategies. These strategies are explained below, where i ($i = 1, 2, ...n$) denotes the current item to be placed, and s is the level created most recently, where the bottom of the object is level 1 created at the beginning of an algorithm [16]. These strategies are explained below and are shown in Figure 2.1.

- **Next fit strategy:** Item i is packed on level k left justified, that is, it is placed at the left-most feasible position if it fits. Otherwise, a new level ($k = k + 1$) is created and i is packed on it left justified.
- **First fit strategy:** It is checked to see whether or not item i can fit from level 1 to level k, and pack it left justified on the first level where it fits. If no level can accommodate i, it is placed on a new level as in the next fit strategy.
- **Best fit strategy:** Item i be packed left justified on the level that minimizes the unused horizontal space among those where it fits. If no level can accommodate$i$, it is placed on a new level as in the next fit strategy.

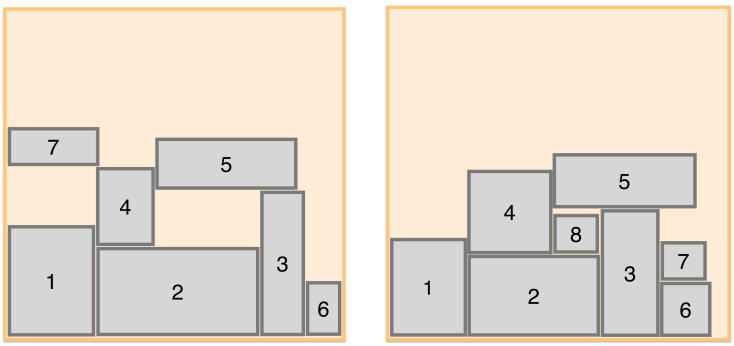


(a) (b) (c)

Figure 2.1. (a) Next-fit, (b) First-fit, and (c) Best-fit strategies.

Baker et al. [17] introduced the Bottom-Left (BL) heuristic, which sorts the items by their areas and packs the current item at the top and then pushes in the lowest possible position, left justified, as shown in Figure 2.2. This method was improved by Chazelle [18] in which, items are sorted according to decreasing width, and each object is located at the most bottom and left place possible, through finding a valid packing position for the item in the hole. Similar to Chazelle's method, Hopper and Turton [19] introduced the Bottom-Left-Fill (BLF) heuristic, which allows for placing each item at the lowest available position in the packing. It is based on the allocation of the lowest sufficiently large region in the partial layout, which enables filling existing gaps in the layout that BL heuristic would have caused. BLF is shown in Figure 2.2. Hopper and Turton [20] presented Bottom-Left-Decreasing (BLD) as an improvement of the strategy of BL, where the objects are ordered using various criteria, such as height, width, perimeter and area and the algorithm selects the best result

obtained. Burke et al. [21] have proposed Best-Fit (BF) which uses a dynamic ordering for the rectangles to be placed by going through the available places from the most bottom left one, and selecting for each place the rectangle that best fits in it. It is used for hybrid meta-heuristic approaches [22, 23].



(a) (b)

Figure 2.2. (a) Bottom-Left heuristic, (b) Bottom-Left-Fill heuristic.

The 2D rectangular packing problem belongs to a subset of classical cutting and packing problems, which has been shown to be an NP-hard problem [24]. Therefore, various approximations or heuristic algorithms have been proposed for solving these problems. The 2D single-bin size packing problems are also studied under 2D container loading or 2D strip packing problems in the literature. Many researchers have used several metaheuristics such as Simulated Annealing (SA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Tabu Search (TS), Greedy Randomized Adaptive Search Procedure (GRASP), and hybrid evolutionary algorithms with heuristics in order to solve nondeterministic polynomial time (NP)-hard problems. An overview of the 2D and 3D single-bin size rectangular packing problems is presented in the following sections.

2.1.2. Simulated Annealing

Simulated Annealing (SA) [25] is an iterative search method. Dowsland [26] used SA for the single bin packing problem, which allows for overlapping pieces. The objective in his study is to reduce the overlap between the pieces down to zero. The objective function is the pairwise sum of the overlap between pieces. The algorithm ends when a layout with

a zero overlap is found or when the number of total iterations are reached. The neighboring solutions are the solutions that are obtained by moving the position of one of the pieces, that contributes a non-zero value to the objective function. An initial strip height is found by packing the pieces according to the BL placement policy with the random ordering of pieces or sorted according to the length, width and area. The minimum height found by any of these orderings is used as the initial strip height, and the items are placed according to that ordering. For annealing, the items that have a contact point with the minimum height are moved down one unit and the height available is reduced and the process repeated.

Hopper and Turton [20] hybridized SA, GA, Naïve Evolution (NE), gradient search (steepest-ascent hill-climbing) and random search (RS) with the BL and BLF heuristics. The fitness function is a weighted sum of the height of the packing items and the packing density, as described in [19]. Rotation changes are applied with a certain probability. Their GA method is the same as they described in [19]. In their approach, NE has the same structure as GA, except it does not apply crossover. SA structure is represented by a permutation of the order of items to be packed. The neighborhood structure of the current solution is found by two methods: by swapping two randomly selected items in the permutation, and changing the orientation of one item. Only one of these methods is applied at an iteration by 0.5 probability.The hill-climbing method has the same neighborhood structure as SA. The initial solution is generated randomly. The search process is stopped after a determined number of unsuccessful moves in the search space. In RS, the permutations are generated randomly. The search process is conducted with the same number of function evaluations as the GA. They found out that the initial ordering of the input sequences according to decreasing width or height of the items improves the outcome. In their experiments, BLF heuristic outperforms the BL routine, however, the time required to place one item is higher for the BLF than BL. In all the problem categories, SA achieves the best layout quality, however, its execution time becomes larger with increasing problem size. GA and NE, have lower execution times and they yield results, which are slightly worse than the ones obtained by the SA. SA, GA, and NE are better in terms of the results than RS and hill-climbing algorithms, due to their ability to escape from local minima. They state that if the time for solving a packing task is limited, GA and NE are appropriate to use since they yield good results and lower executions times than SA.

Dereli and Daş [27] propose a hybrid SA with a recursive placement procedure, in which items are packed to the BL available subspaces of the previously placed items. The neighbor solutions are formed by the mutation operator, which rotates an item. After the mutation, the items are sorted in decreasing order of height and the sorted items are used for the recursive placement procedure. They test their hybrid-SA algorithm on a number of test cases taken from the literature, compared the results to GA+BL, GA+BLF, SA+BL, SA+BLF [20] and Heuristic Recursive (HR) [28]. The computational results show that the proposed SA algorithm outperforms GA+BL, GA+BLF, SA+BL, and better than SA+BLF and HR in some cases.

Egeblad and Pisinger [29] used SA for the 2D and 3D rotation-allowed knapsack packing problem. In their approach, for the 2D packing, a small modification to the sequence pair, transform the sequence pair into a packing, evaluate the profit of the corresponding packing and accept the solution depending on the outcome. The neighboring solution is chosen randomly to be one of the methods through exchanging two items in one of the sequences, exchanging two items in both sequences, and changing the rotation of an item. The initial configuration is found by sorting items according to non-increasing efficiency and filling the knapsack in a greedy way until the first item does not fit into the knapsack.

Peng et al [30] propose a hybrid SA for the 3D container loading problem. They use block structure for representing boxes of the same type with same the spatial orientation, which are placed at the left-rear-bottom corner. Given a packing space, a box type, and its orientation, a greedy filling algorithm is used to generating blocks. The neighboring solution is created by changing only one item at a time. For the initial configuration, a random packing sequence is used. They use volume utilization as the objective function. Their proposed algorithm outperforms some methods in the literature, namely Bischoff et al's [31, 32] and Gehring and Bortfeldt's [33–35].

Dereli and Daş [36] proposed hybrid SA algorithms for single and multi-objective container loading problems, in which they use a wall-building procedure for the packing similar to the procedure proposed by George and Robinson [37]. For filling each layer, the dimensions of each layer are determined by choosing the width of the layer through

the layer-determining item. Their objective function is a weighted sum of deviations from the weight of the items and volume utilization goals. The structure is represented by a bit string representation, in which bits represent the rotation of the item types. The neighboring solution is found by flipping the value of four of the item types, which are randomly selected from the solution.

Wang et al [38] used the SA algorithm for the 3D container loading problem. They use a three sequence structure to represent a solution: box type packing order sequence, layer type sequence, and layer position sequence. First-fit empty maximal-space list is used to manage the free spaces within the container and difference process is applied in the space decomposition. A neighboring solution is created by randomly choosing between exchanging two elements of box packing order sequence and modifying one element of layer type sequence. The fitness function is the ratio of packed volume to the total volume of the container.

Can and Şahingöz [39] study the 3D container loading problem with three type of box sizes, namely, small, medium and large boxes. The initial ordering for packing the items is generated randomly. They assume that the container can contain all the packable items, therefore, the objective function is minimizing the unused space volume in the container. The neighboring solution is created by randomly swapping two elements of the item queue.

2.1.3. Genetic Algorithm

Genetic Algorithm (GA) [40] is a search heuristic that imitates the process of natural selection, by the inheritance, mutation, selection, and crossover techniques. Kröger [41] studied guillotinable bin packing problems by using GA and SA. He uses the packing density within the bin for the fitness function. Slicing-tree and hill-climbing methods are used for building the structure of the chromosome. He uses subtrees for the mutation and the crossover operations. He compared SA and GA methods, and found that GA had higher fitness than SA. Jakobs [42] proposed a GA for the 2D single-bin packing of rotatable items problem that uses BL heuristic, in which the number of items is less the number of possible packable items, hence, the fitness function is the contiguous remainder area in the bin.

He defines the chromosome as the ordering of the item numbers, i.e. genes, according to BL-heuristic placement. The initial population is created by one individual which has non-increasing width order in its chromosome, and the rest of the individuals have a random order in their chromosomes. The chromosomes are sorted in the population according to their fitnesses, and two random chromosomes, i.e. parents, are selected. For the crossover operation, a random position is selected at the first parent and a random number of items are moved from the first parent to the child and the rest is filled from the second parent, which is a similar approach to the two-point crossover. Two types of mutations are applied to the child chromosome (offspring): a rotation mutation and inversion. Afterward, the items are placed in BL-order, and the fitness is calculated. The individual with the worst fitness is replaced by the new chromosome.

Gehring and Bortfeldt [33] proposed a hybrid GA for the 3D single-bin packing, i.e. container loading problem, with rotation. In their approach, the chromosome is coded according to the order of the items. In order to keep a fixed size chromosome, they use a virtual container that has infinite size, so that it would contain all the items. A feasible solution has a subset of the items in the chromosome that are in the boundaries of the real container. The chromosome order is Bottom-Left, that is, the items that are at the bottom level in x, and left-most in y come prior than other items. Tower sets are used for finding the placement. The chromosomes are sorted in the population, according to the objective function and the fitness is found after ranking, and the chromosomes with the highest fitness values are selected for the crossover or mutation operations. The crossover is a uniform order-based crossover, in which two offsprings are created, and the two chromosomes with the lowest fitness values are replaced. Scramble sublist mutation is used for replacing the individual with the lowest fitness value. The crossover and mutation have 0.5 probability, that is, at each iteration only one of them is applied.

Hopper and Turton [19] proposed a hybrid GA using the BLF heuristic for the bin packing problem which consists of a fixed number of items, that are packed into a single bin of fixed width and unlimited height. They use partially matched crossover (PMX) and order-based mutation. A second mutation operator is used for changing the orientation of each rectangle in the sequence with a certain probability. The fitness function is a weighted sum

of the height of the packing items and the packing density. The initial population is formed by sorting rectangles according to decreasing height. They use “elitist strategy”, which uses a portion of the previous population for generating the current population. In their study, they found that pre-ordering the input according to decreasing height improves the outcome. Their GA with the BLF heuristic outperforms the GA using the BL method, as well as the BL and BLF methods on their own. However, they state that the difference between their hybrid GA and the BLF heuristic is not very significant and that an improvement on the GA encoding schemes might change this.

Gehring and Bortfeldt [43,44] proposed another hybrid GA, with the addition of layer-type stowing plans and different generation formation methods. A layer is determined by the size of the bottom item, which is called the layer-determining piece, and its orientation. The size of the layer-determining piece in x determines the layer depth. An item can be packed into that layer if it has a size that is less than the layer depth. The layers are essentially columns in the placement, and each layer can have a different depth. The layers represent the genes in the chromosome. The fitness function is the covered part of the container area. “Elitist strategy” is used for creating new individuals by reproducing ten best individuals from the previous population. The rest of the population is formed by crossover with 0.66 probability, and mutation with 0.33 probability. One of the parents is chosen randomly for the crossover and the other is chosen through the ranking of the individuals depending on their relative fitnesses, as described in [33]. In order to create an offspring, the layers of both parents are sorted in descending order according to the fitness. The best parent layer is transferred to the offspring for the crossover. The other parent layers are transferred only if their depth does not exceed the container length and if the total number of existing pieces per type is maintained. For the mutation, the number of layers to be transferred are selected at random, at maximum 50% of the parent layers may be transferred. After the generation of the current population, ten merging mutations are applied to the current population, in which all but two the layers of the parent are transferred to the offspring. The offspring of the merging mutation replaces the worst individual of the population in terms of the fitness if it has a higher fitness. In the crossover and mutations, additional items are placed in the layers if there are any spaces left within the container. The population size is 50 and 500 generations are created. Duplicate chromosomes are not allowed in the population. The

ending condition is determined by the number of generations a placement or if all the available items are packed, the search is aborted. Gehring and Bortfeldt [43] proposed a parallel GA with the previous methods introduced. In their study, they parallelize the formation of the populations, and use different “migration topologies”, “migration strategies”, “migration interval” and “migration intensities”.

Wu et al [45] proposed a GA for the variable bin height. In their approach, the gene consists of the item number and the rotation of the item. The initial population is formed by sorting the items according to volumes and randomly assigning rotation positions to each item for each chromosome. Roulette wheel selection and niche mechanism are applied for reproduction of 10% of the population. The crossover operation is one-point crossover, where the crossover point is random. The duplicates are eliminated and replaced with the missing items or random items after the crossover. Two types of mutation are applied: inversion mutation in which two genes are swapped, and random rotation mutation. They use First-Fit Decreasing for the placement of items.

Pintea et al [46] compared several heuristics for the 2D bin packing problem. They represented their genes with the item numbers. The fitness is the maximum height of the placed items. “Elitist strategy” is used for reproduction. Inversion mutation and one-point crossover with random crossover point are used. In the initial population, the first individual has a sorted-size ordering. In their tested datasets, sorting the rectangles in an ascending order of heights give the best solution. BLF heuristic is used for placing the items.

2.1.4. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [47] is a search heuristic that uses a set of particles called a swarm, and move them with a variable velocity in the search-space to find a solution. The velocity depends on the best position of the particles and the overall best position of the swarm. Zhao et al. [48] applied discrete PSO for solving the bin packing problem, by using the remainder rectangular packing rule for the placement of items. The location of the items is determined through the packing rule. The speed in their approach determines the exchange of particle locations, that is, the items are exchanged locations similar to SA. The utilization

ratio of the container is the fitness value.

He et al [49] also used discrete PSO for solving the 3D bin packing problem. They have three fitness functions: maximizing the volume of the items that are packed, maximizing the weight of these items and maximum stability. They use three-space partitioning algorithms for packing. They define the position of the particle by the ordered set of edges, and the calculation of velocity is suited to their approach through the use of edges.

Yang et al [50] presented a PSO algorithm for the 2D rectangular packing problem. In their approach, they use the attractive factors routine for packing, in which items move through the attraction by the attractive factors, in order to fulfill the layout. The sequence of the items is in sorted order by the area of the rectangles. The location of a particle is represented by a 7-dimension vector of the weights of the attractive forces. The fitness function is the utilization ratio of the rectangular region. Particles are assigned a random velocity and location. They use a perturbation strategy, in order to prevent PSO from being stuck at local optima, in which one of the eliminate-inferior-particle routine, the restart routine, the vary-particle routine and the quasi-GA is selected randomly.

Shin et al [51,52] used original PSO and improved PSO algorithms in their study for the continuous-value optimization problem in bin packing for arbitrary polygon-shaped packing regions. They propose an improved PSO algorithm, which uses a second best particle in the swarm, in order to avoid being stuck at local optimums. For the bin packing problem, they search for the number of items that can be packed into the polygon by adding a new item at each iteration in a predetermined order and iterate PSO and improved PSO algorithms for the determined number of cycles. The coordinates of the items are randomly assigned at the beginning of every iteration. The best particle in the swarm and the particles' best positions are reset at the beginning of each iteration for adding a new item. The fitness function is determined by the number of overlaps and the number of items outside the boundaries of the polygon. If every item is inside the polygon, and there are no overlaps, then the fitness is 1, otherwise, it is less than 1. If at the end of the iteration for PSO algorithms, the best particle has a fitness value less than 1, it means that there are overlaps even in the best placement for the current number of items, therefore, the execution ends with the previous number of items

that has a non-overlapping solution.

Domingo et al [53] proposed a PSO algorithm for the 3D container problem, in which each item is represented by a key. The keys are randomly assigned to the items at the beginning in the [0,1] range. Afterward, the items are sorted according to these keys. The sorted order determines the input sequence of items to the packing scheme. The keys are changed at each iteration through the velocity component, which depends on the particle's best key, which results in the highest fitness and the best particle's key, that has the best fitness overall. The packing scheme is the layer building approach with the back-bottom-left procedure. The fitness for each particle is found by its volume utilization in the container. The PSO algorithm is applied to this structure, to find the placement which has the particles with the highest volume utilization.

Chen et al [54] propose a rectangular layer-packing algorithm (RLPA) which uses layers of items for the packing and apply their packing method to PSO and GA for the 2D rectangular layer-problem without rotation. In their approach, a particle is encoded as a packing scheme, i.e. the list of items that are packed through RLPA in the solution. They define a fitness function that depends on velocity of each particle, the overall width and height of the container, the width and height of each item in a layer. For GA, they use a very similar fitness function. The chromosome consists of a permutation of numbers, which represent a packing scheme, similar to PSO. Initial population is generated randomly. A relative fitness is used with a random number is used similar to [42], to select a parent. For the crossover, they generate two crossover points, randomly. If the first number is less than the second number, they use the genes between these two numbers for the offspring. Otherwise, the genes that are outside this range are used for the offspring. The mutation operator randomly selects some determined number of genes from the parents and changes them. They find that GA gives a more optimized solution for the problems, while PSO takes less time for creating a good solution.

2.1.5. Tabu Search

Tabu Search [55] is a search method that employs local search methods. Bortfeldt and Gehring [35] proposed a parallel TS algorithm for solving the container loading problem with a single container for the weakly heterogeneous load case. They use a block structure which represents boxes of the same type with same spatial orientation. A local arrangement consists of one or two blocks, which are called 1-arrangement or 2-arrangement. In each block, the box numbers in all three dimensions are initially determined such that the dimensions of the packing space are utilized as fully as possible. For the objective function, maximizing the total volume of the boxes stowed in the packing space is considered, along with minimizing loss volume and maximizing effective volume. They use two types of neighboring solutions, which are determined through a heuristic. The parallel searches are carried out with differently configured instances of a TS for finding the best solution.

Gendreau et al [56] use TS for the vehicle routing and 3D container loading problem. In their approach, the items are initially sorted such that all the non-fragile items precede the fragile ones, and that each of these two subsets is sorted by decreasing volume, and if two items have the same volume than through decreasing height. For the first sequence, the items are listed in the inverse order. The algorithm uses two tabu lists, one per item type. The interchange two items from different lists is a tabu. However, this is accepted when it produces a solution improving on the objective. The search terminates when a feasible loading is found or after a prefixed number of iterations. They use the BL heuristic with the touching perimeter algorithm [57] for the packing.

Viegas et al [58] present a TS with best-fit decreasing heuristic for the real-world steel cutting problem. retail steel distributor. Their objective function is the minimization of stock growth. They use a 3D heuristic algorithm, similar to first-fit decreasing heuristic.The open guillotine cut algorithm is used after the placement of each box in order to find the cutting pattern which maximizes the dimensions. The element range is used for the algorithm to bound the neighboring solutions in order to find feasible solutions, in which only one element is different from the previous solution. The algorithm stops after a fixed number of iterations without improvement.

2.1.6. Greedy Randomized Adaptive Search Procedure

Greedy Randomized Adaptive Search Procedure (GRASP) [59] is a metaheuristic search algorithm, which consists of successive constructions of a greedy randomized solution and subsequent iterative improvements of the solution through a local search. Moura et al [60] introduce GRMODGRASP, which is based on the improved wall-building heuristic proposed by George and Robinson [37], called GRMOD. The differences between GRMOD and the wall-building heuristic are the restriction of the container to be a finite-length container, and the layer depth dimension depends on the volume of the unpacked boxes, such that the final layers of the container have a smaller depth but better volume utilization. For creating solutions in the GRASP, the algorithm loads the container until it meets a stopping condition: there is no free space in the container, all the boxes are packed, or the remaining free space is smaller than any of the unpacked items. After choosing an empty space to pack an item, the algorithm chooses the next type of box to pack from a candidate list, where the boxes ordered by their volume utilization.

Parreño et al [61] propose a GRASP algorithm with a constructive block heuristic that uses the concepts of maximal-space, and a non-disjoint representation of the free space in the container. A block is a rectangular packing of boxes of the same type, in rows and columns, for filling only one side of the empty space. The feasible placement positions are in the form a list of empty spaces and each time a block is placed in an empty space, new spaces are added to the list. The procedure of the GRASP algorithm consists of the constructive phase through the block heuristic and improvement phase through eliminating some blocks from the solution from the last items that are packed and filling the empty spaces with the deterministic constructive algorithm. The largest increase in the volume used and the best fit into the empty maximal-space are considered as the objective function. They improve their algorithm for the multi-drop constraints in [62].

2.2. Object Recognition

RGB-D sensors significantly facilitated object recognition [63] for the use of service robotics applications, thus enabling the execution of tasks, such as two-handed folding of

towels [64] and clothes [65] or cutting fruits and vegetables [66]. A combination of object recognition methods presented below is used in several robotics applications for manipulating household objects [67–71].

For the 3D object recognition problem, the phases are described in the survey of Guo et al [72] as follows:

- *3D Keypoint Detection*: Points with defining information content are identified as keypoints. The location and scale of a keypoint define a local surface which is used in the subsequent feature description phase
  (i) Fixed-Scale Keypoint Detection: This type of keypoint detection defines a keypoint to be a point, which is distinctive within a predetermined region. There are two types of methods: curvature based methods and other surface variation measure based methods.
  (ii) Adaptive-Scale Keypoint Detection: These methods work by initially building a scale-space for a given range image, and then choosing the points that have high distinctiveness in both the spatial and scale neighborhoods as keypoints. For this type of keypoint detection several types of methods are used in the literature, such as, coordinate smoothing based methods, geometric attribute smoothing based methods, surface variation based methods, transform-based methods.
- *Feature Extraction*: Geometric information of the local surface around a keypoint is extracted and encoded into a representative feature descriptor.
  (i) Signature Based Methods: These methods use geometric measures that are computed individually at each point in the neighborhood. Examples of descriptor types are Binary Robust Appearance and Normal Descriptor, Log-Polar Height Map, Normal Aligned Radial Feature.
  (ii) Histogram Based Methods: These methods use histograms for accumulating geometric or topological measurements to describe the local neighborhood of a keypoint. There are three types of histogram based methods: spatial distribution histogram, geometric attribute histogram, and oriented gradient histogram.
  (iii) Transform Based Methods: These methods transform a range image from the

spatial domain to another domain and describe the neighborhood of a given point through the information in the transformed domain. Heat Kernel Signature and 3D Speeded Up Robust Feature are these types of methods.

- *Surface Matching*: Surface matching is conducted in three steps: Feature matching, hypothesis generation, and hypothesis verification.
  (i) Feature Matching: Scene features are matched against all model features in the library, resulting in a set of feature correspondences and hypotheses. Most popular strategies for feature matching are threshold based, nearest neighbor (NN) based and nearest neighbor distance ratio based.
  (ii) Hypothesis Generation: Feature correspondences are used to vote for the candidate models and generate transformation hypotheses. The methods used for hypothesis generation are: geometric consistency, pose clustering, constrained interpretation tree, Random Sample Consensus (RANSAC), game theory, generalized Hough transform, and geometric hashing.
  (iii) Hypothesis Verification: Hypotheses are verified to infer the identity and pose of the object. There are two types of hypothesis verification: individual verification methods, such as those using Iterative Closest Point (ICP), and global verification methods

There are also applications developed with *ROS* [73] for object recognition, such as *Object Recognition Kitchen* [74], *RoboEarth* [75] and *MOPED* [67]. *OpenCV* [76] and *PCL* [77] libraries are used for implementing these software and packages.

3. SYSTEM DESIGN

In order to build a system according to our goals in this project, we first defined our design requirements. In this chapter, the hardware and software requirements for our system are presented in Section 3.1. According to these requirements, we have built a system that uses a robot arm, a depth camera, and a computer, which are presented in detail in Section 3.2. The processing flowchart of our system is shown in Section 3.3.

3.1. System Requirements

Our goal in this project is using a real robot for placing mugs on the counter-top to the dishwasher tray. Therefore, there are two types of design requirements for our system: the hardware requirements and the software requirements.

3.1.1. Hardware Requirements

3.1.1.1. Functional Requirements.

- The robot arm must be able to move a mug from the kitchen counter to the dishwasher so it should have at least 5 Degree-of-Freedom (DOF)
- The robot arm should be able to grasp a mug, hence a gripper with at least 10 cm opening capacity is necessary
- The system must be able to lift a mug, therefore a robot arm with minimum payload of 200 g is necessary
- Each joint must be separately controllable, therefore 5 motors with encoders should be used for a 5 DOF arm
- The system must supply the voltage needed for the motors, hence a power source of 12 V DC is necessary
- A depth camera should be used for detecting the objects on the counter-top
- The gripper should not break the mug, therefore a force sensor resistor should be used on the surface of the gripper

3.1.1.2. Nonfunctional Requirements.

- The system shall finish its operation for 1 mug in less than 3 minutes
- The system shall place mugs on the tray with no collisions
- A low-cost system shall be built

3.1.2. Software Requirements

3.1.2.1. Functional Requirements.

- The system should have an operating system for the autonomous loading of the dish-washer tray
- A programming language should be used for the system software
- The system should have a program that controls the motors on the robot arm
- A simulation program should be installed on the system for calculating the kinematics for the arm and detecting objects
- The system should detect the mug and perceive its sizes
- The system should find a near-optimal non-colliding placement solution for the mugs to be placed

3.1.2.2. Nonfunctional Requirements.

- The system shall detect all of the mugs in less than 60 seconds
- The system shall calculate the kinematics of the arm for the pick and place action in less than 60 seconds
- The system shall calculate the placement positions on the tray for the mugs in less than 60 seconds
- The system shall have a ready to use software for the ease of usability
- The software shall be extendable to work for the placement of different objects

3.2. System Components

We designed a system that satisfies our design requirements. A low-cost 5 Degree-of-Freedom (DOF) custom physical robot arm developed in Bogazici University's Robot Lab [78] is used for the manipulation. *ROS Indigo* [73] is used for connecting to the physical robot and *Gazebo 2* [79] is used as the simulation environment. *ROS MoveIt!* program is used as the kinematics engine in motion planning [80]. *ASUS Xtion Pro Live* [81] RGB-D camera is used for detecting and recognizing the objects. The program is run on a computer with 15.7 GB RAM, Intel Core i7-2600 CPU at 3.40 GHz x 8, on Ubuntu 14.04 operating system. The full system is shown in Figure 3.1.

Figure 3.1. Our robot system with 3D depth camera and 5 DOF robot arm.

3.2.1. The Robot Arm

The joints of the arm are shoulder turn (S_T), shoulder lift (S_L), elbow lift (E_L), wrist lift (W_L), wrist turn (W_T), and gripper (G_L), which are all revolute joints, as shown in Figure 3.2. The arm acts like a right arm, in principle. The motors are *Robotis Dynamixel* [82] series servomotors and their frames. The gripper and the wrist turn uses the AX-12A servomotors, and the motors for the shoulder turn and elbow lift joints are MX-64T servomotors. MX-series provides continuous rotation, acceleration, and contactless magnetic encoders. The wrist lift motor is MX-28T, and the shoulder lift motor is MX-106T, which is the strongest of the motors used, because of the torque it needs to provide for the arm is high. The maximum overall payload for the arm is 0.2 kg at 0.728 m away from the shoulder. The overall cost of the arm is $1,788.2 USD [78]. The recommended voltage for the MX motors is 12 V, which is the maximum voltage level of AX-12A, therefore, we are using a 12 V DC power supplier [82]. The motors are connected to the computer through *Robotis USB2Dynamixel* interface, which uses an FT232 chip to enable USB to TTL interfacing [83].

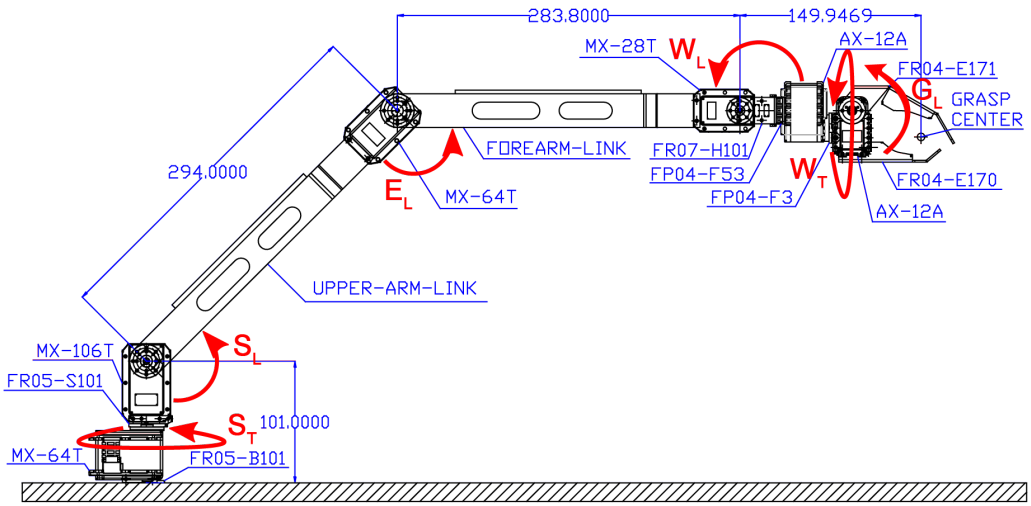


Figure 3.2. Technical Drawing of the robot arm [78].

We have derived the Denavit-Hartenberg (DH) parameters [84] for the kinematics of the arm. a_{DH} is the distance between the axes Z_{i-1} and Z_i, which is measured along X_i. d_{DH} is the distance between the origin O_{i-1} and the intersection of the X_i axis with Z_{i-1},

the distance is measured along the Z_{i-1} axis. α_{DH} is the angle between the axes Z_{i-1} and Z_i, which is measured in a plane normal to X_i. α_{DH} is calculated from Z_{i-1} to Z_i by the right-hand rule. θ_{DH} is the angle between X_{i-1} and X_i, that is measured in a plane normal to Z_{i-1}. θ_{DH} corresponds to the joint angle for a revolute joint [85]. The kinematic chain of the arm and the respective DH parameters for each link are presented in Figure 3.3 and Table 3.1. The a_{DH} and d_{DH} are expressed in meters.

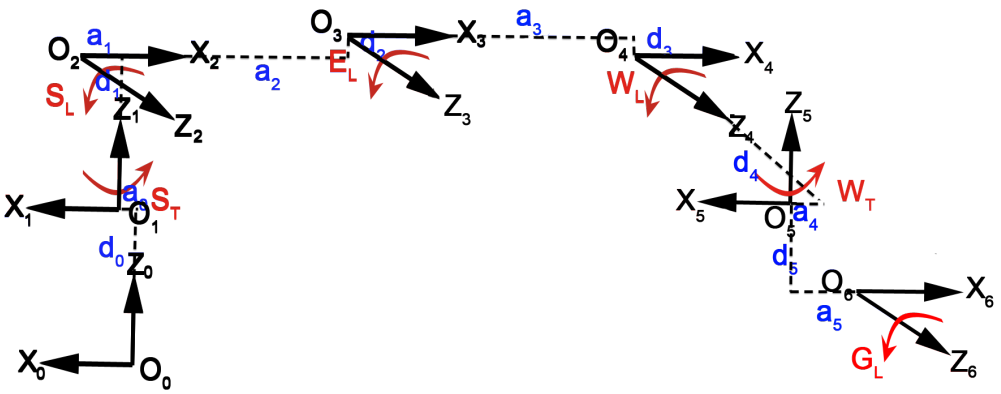


Figure 3.3. Denavit-Hartenberg assignment for the 5 DOF robot arm.

Table 3.1. Denavit-Hartenberg parameters of the arm.

Link	a_{DH} (m)	d_{DH} (m)	α_{DH}	θ_{DH}
0	-0.000075	0.01	0	S_T
1	-0.000465	-0.00000009	$-\pi/2$	S_L
2	0.294	0.00000009	0	E_L
3	0.2838	-0.00000003	0	W_L
4	-0.00003	0.074	$-\pi/2$	W_T
5	-0.00001	0.019	0	G_L

The robot arm is intended to be used as a part of another project, *TURGAY*, in the Bogazici University Robotics Group, which designed as a tour guide robot to be used in museums [86]. The *TURGAY* robot has a *Festo Robotino* base [87], a metal custom body,

Hokuyo URG-04LX laser scanner [88] and a *ASUS Xtion Pro Live* [81] RGB-D camera. For the purposes of this project, the arm, and the camera were detached from the robot.

A *Pololu* brand force-sensing resistor (FSR) [89] is used to recognize a grasp: when the gripper is to be closed, the gripper opening is decreased until the analog value read from the resistor is above a certain threshold for a firm grasp, which was determined empirically. If the value is above the threshold, the grasp is successful, otherwise, if the gripper aperture is very small, which would not allow for a mug to be held, the grasp is unsuccessful. The force-sensing resistor is a 4.37 cm x 4.37 cm square, with 3.81 cm x 3.81 cm force sensing range. It weighs 1.2 g, hence, the effect of it on the payload of the gripper is negligible. It is glued to the inner pad of the gripper. An *Arduino Uno* is used to obtain the values from the force-sensing resistor with a 10 kΩ pull-down resistor and 5 V voltage supplier pin as a voltage divider. The *Arduino* is located at the base of the arm, and it is powered by the USB port from the computer. FSR is also used to detect if the mug fell from the gripper during any part of the manipulation.

3.2.2. Robot Operating System

Robot Operating System (ROS) [73] is an open-source robotics framework, that has multiple libraries and tools that allow building robot applications. It can be run on a computer or on some robot systems. The supported operating system for *ROS* is *Ubuntu*. The most recent version is *ROS Jade*, however *ROS Indigo* is recommended for stability. We use *ROS Indigo* in our project for this reason. *ROS* enable developers to implement nodes and message-passing functionality in C++, Python, and LISP programming languages, and it also has experimental libraries written in Java, Haskell, Go, Julia, and R. We are using C++ language for the manipulation and placement planning in our system. The object recognition part is written in Python [90].

The initial *ROS* integration of the robot arm was made by N. Ergin Özkucur and Ayberk Özgür. We modified the code to use *ROS MoveIt!* as the kinematics engine and the *Kinematics and Dynamics Library* for calculating the inverse kinematics for the motion planning. We also integrated the libraries necessary to use the *Gazebo* simulation environment.

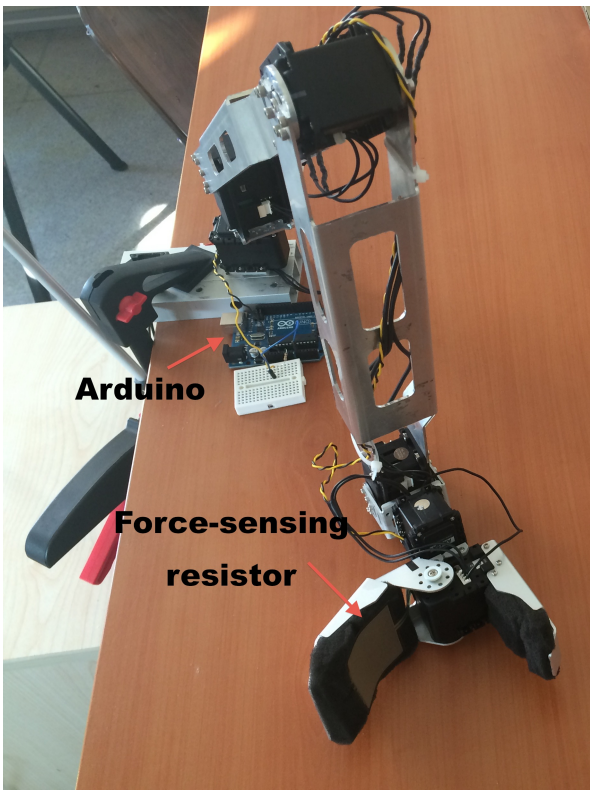


Figure 3.4. Force sensing resistor mounted to gripper.

ROS MoveIt! [80] is an open-source software that incorporates motion planning, manipulation, 3D perception, kinematics, control, and navigation. It requires a *Unified Robot Description Format* (*URDF*) of the robot, which is an *XML* format for representing a robot model [91]. The *URDF* contains coordinates, mass, inertia, collision information and visual *Stereolithography* (*STL*) file for each link and joint in the robot. *STL* files describe the surface geometry of a 3D object without color, texture or other computer-aided design (CAD) model attributes [92]. The *STL* files were created in BRL-CAD [78, 93]. *MoveIt! Setup Assistant* was used to create the "planning group" object, defined as the "move_group". Initially, a *Collision Matrix* is generated through the *URDF*. The adjacent links are defined as "never in a collision", in order to separate them from the other stationary or movable links that could collide if they touch each other. This allows the *ROS MoveIt!* to define the "valid" moves for the arm. A move is "not valid", if any of links collide in that move or if a target position is impossible for the robot to reach. A planning group "arm" is created to control the joints of the arm through the kinematics. The "arm" consists of the chain of links starting from the "base_link", which is the shoulder link, to the link of the wrist turn joint. It also includes the shoulder turn, shoulder lift, elbow lift, wrist lift and wrist turn joints. The "gripper" planning group was also created for controlling the gripper apart from the "arm". The "gripper" group contains the fixed and the movable link of the gripper and the gripper joint. The "end effector" is chosen as the "gripper". The "parent link" of the "end effector" is the final link in the "arm" group, and the "parent group" is the "arm".

The *Kinematics and Dynamics Library* (*KDL*) [94] is chosen for the "arm". However, *KDL* only supports 6 DOF arms, which results in wrong kinematics calculations for our arm for the side grasps, because of the missing wrist yaw joint. Hence, we used *MATLAB R2014a* [95] for fitting an error function using the curve-fitting tool. The error function changed for each z position of the end effector, where z is the coordinate perpendicular to the floor. The error functions and their derivations are presented in Appendix A. The "gripper" group does not require a kinematics library.

The robot poses are determined through the *MoveIt! Setup Assistant* for the "arm", which are the positions of the arm to grasp from the side, grasp from the top, and the default pose in which all the joints are set to zero. The set poses for the "gripper" are the two set

angles of the gripper, in which the gripper is open and it is closed. These poses enable comparing the angles created through the *MoveIt!* in RViz [96], which is a 3D visualization tool for ROS, to the real angles necessary for the robot.

The *MoveIt! Setup Assistant* generates the configuration files, determined through the steps mentioned above: a *Semantic Robot Description Format* (*SRDF*) [97], which contains the semantic information about the robot, the *controller*, *joint_limits*, *kinematics* and *ompl_planning* YAML [98] files and the launch files for loading the parameters necessary for controlling the robot in a simulation environment or in RViz.

3.2.2.1. ROS Modules. The *ROS* modules used in this project are:

- *tf*: This package allows the user to keep track of multiple coordinate frames over time. The relationship between the coordinate frames is maintained in a tree structure buffered in time, hence, allowing the user to transform points and vectors between any two coordinate frames at any desired point in time. This package is used in our system for transforming the object poses that are relative to *ASUS Xtion Pro Live* received from the object recognition node to the world coordinate frame, i.e. the frame of the base link corresponding to *Festo Robotino*. The target position for the arm is found from the derived object coordinates relative to the world frame. The target position is transformed to the "end effector" frame, in order to find the kinematics of the arm to reach the target position, through the methods of *ROS MoveIt!* and *KDL* libraries.
- *Messages*:
  - (i) *std_msgs*: These are the standard *ROS* messages including common message types representing primitive data types and other basic message constructs. We are using Header, Int32, Float32, Float64, Uint8, Uint32, and String data types in our system.
  - (ii) *geometry_msgs*: This library contains messages for common geometric primitives such as points, vectors, and poses. In our system, Pose and PoseStamped, are used to define the 3D poses of the mugs detected on the countertop.
  - (iii) Custom messages: We have defined our own messages for the object detection in

order to pass the information between nodes. **ObjectDetection.msg** consists of time and relative coordinate frame information, id of the object, confidence value of the object, pose of the object: pose.position, pose.orientation, and sizes of the object. **ObjectDetections.msg** contains the list of all objects detected.

- *Services*: A ROS node provides a service under a string name, and a client calls that service by sending the request message and awaiting the reply. Services are defined using *srv* files, which turn into header files for use in C++ when compiled. We have custom services in our system:
  (i) **arm_isvalid.srv**: This service sends the actual target pose, and the target pose of the arm, which is corrected by the error function for the kinematics, as shown in Appendix A. It also sends the method of calculation for the kinematics, i.e. through a trajectory of positions or through continuous iteration. The result gives if target a position is a "valid" position, such that the kinematics can be calculated within an error margin stated in Appendix A.
  (ii) **arm_planner_config.srv**: This service is used for opening and closing the gripper within a continuous mode to be used with a force sensing resistor as explained in Section 4.3.1. The result is the state of the grasp of an object, i.e. successful or not.
- *actionlib*: This package allows canceling actions that take a long time to execute or getting periodic feedback about how the service request is progressing. In our system, we are using the service that allows the kinematics of the arm to be calculated and executed at a different node than the node that calculates the target position for the arm. Therefore, the service request for the execution of the arm kinematics has to be checked in the main node, in order to send the next target goal, if the execution of the kinematics of the current target position has finished. We wrote a custom action file for the kinematics, as follows:
  (i) **MoveArm.action**: This action file transmits the poses from the placement planning node to the kinematics node. The a predefined pose index the predefined pose for the arm, if such a pose exists. The target, the actual target poses, and the use of trajectory planning are the same as stated before. A previous trajectory can be used, such that the calculation of the kinematics for that trajectory takes

less time, as it is calculated before. Also, the state of the gripper rotation action is sent through this file, which represents if the gripper is to be turned upside-down. In such a case, the action kinematics is not calculated, since only the wrist turn joint is turned π degrees. The result of the state of this action, that is, if the action was successful or not, is returned as a feedback to the planner.

- *openni_launch*: This package contains launch files for using *ASUS Xtion Pro Live* RGB-D camera. Launching this package creates a nodelet graph to transform the raw data from the camera driver into point clouds, disparity images, and other products suitable for processing and visualization.

We use *dynamixel_motor*, *dynamixel_controllers*, *dynamixel_driver* packages [99], which enable controlling the Robotis Dynamixel servomotors and setting their speed, torque and proportional–integral–derivative controller (PID) values through *ROS*.

3.2.2.2. Force-Sensing Resistor with Arduino. FSR value is read continuously from *Arduino* [100], which publishes the data as a ROS topic, and if the value is decreased below the grasp threshold then the current state changes to the object detection stage and the current mug is searched. If the grasp is unsuccessful during the grasping action, the same state change occurs. After the current mug is found, the position of the mug is updated and the state changes to the initial starting position and the mug is placed on the empty area again to be grasped from the side.

3.2.3. Simulation Environment: Gazebo

In order to test our system and find the limits of our arm we are using a simulation environment, *Gazebo 2* [79], which is supported by *ROS Indigo* as the package *ros-indigo-gazebo*. *Gazebo* is a 3D dynamic simulator that can accurately and efficiently simulate robots in complex indoor and outdoor environments. It allows to create "world" files, which is defined by the physical laws of the simulation environment, such as the gravity and friction coefficients of the surface. The parameters for the objects in the simulation environment consist of their 3D poses, linear and angular velocity, acceleration, inertia, mass and their

sizes and shapes. The objects can be imported from the *Model Database* or custom objects can be created through defining these parameters and by providing STL files for the visuals.

We are using *Gazebo* as the simulation environment, because it is compatible with *ROS MoveIt!*. We are using additional *Gazebo* plugins *gazebo_ros_control*, which connects *Gazebo* to *ROS* and *kinect_camera_controller* for using a depth camera for detecting the virtual objects in the *Gazebo* environment. The robot is modeled with a mobile base in the simulation environment because the arm is intended to be used as a part of the *TURGAY* project, as noted before. Hence, the global world frame is defined at $(x, y, z) = (0, 0, 0)$ coordinate in *Gazebo*, which corresponds to the floor, instead of the arm base. Figure 3.5 shows the robot and our experimental setting in *Gazebo*.

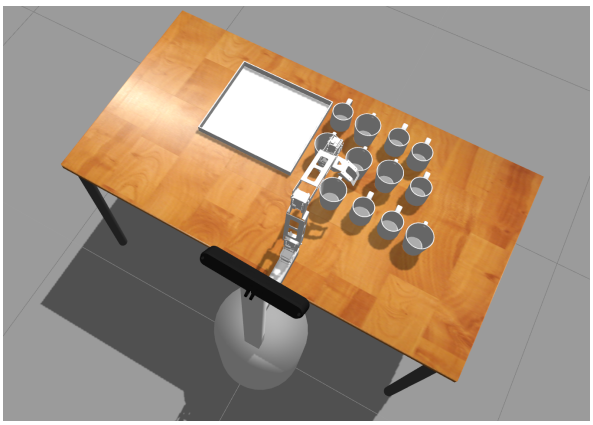


Figure 3.5. Our robot system in *Gazebo*, simulation environment.

3.3. System Outline

Our system consists of object recognition for detecting the mugs on the counter-top, placement planning for optimizing the placement of mugs within the tray and manipulation planning for placing the items on the counter-top to the dishwasher tray. The flowchart of the processing of our system is given in Figure 3.6. Furthermore, the detailed processing flow of the placement planning is presented in Figure 3.7.

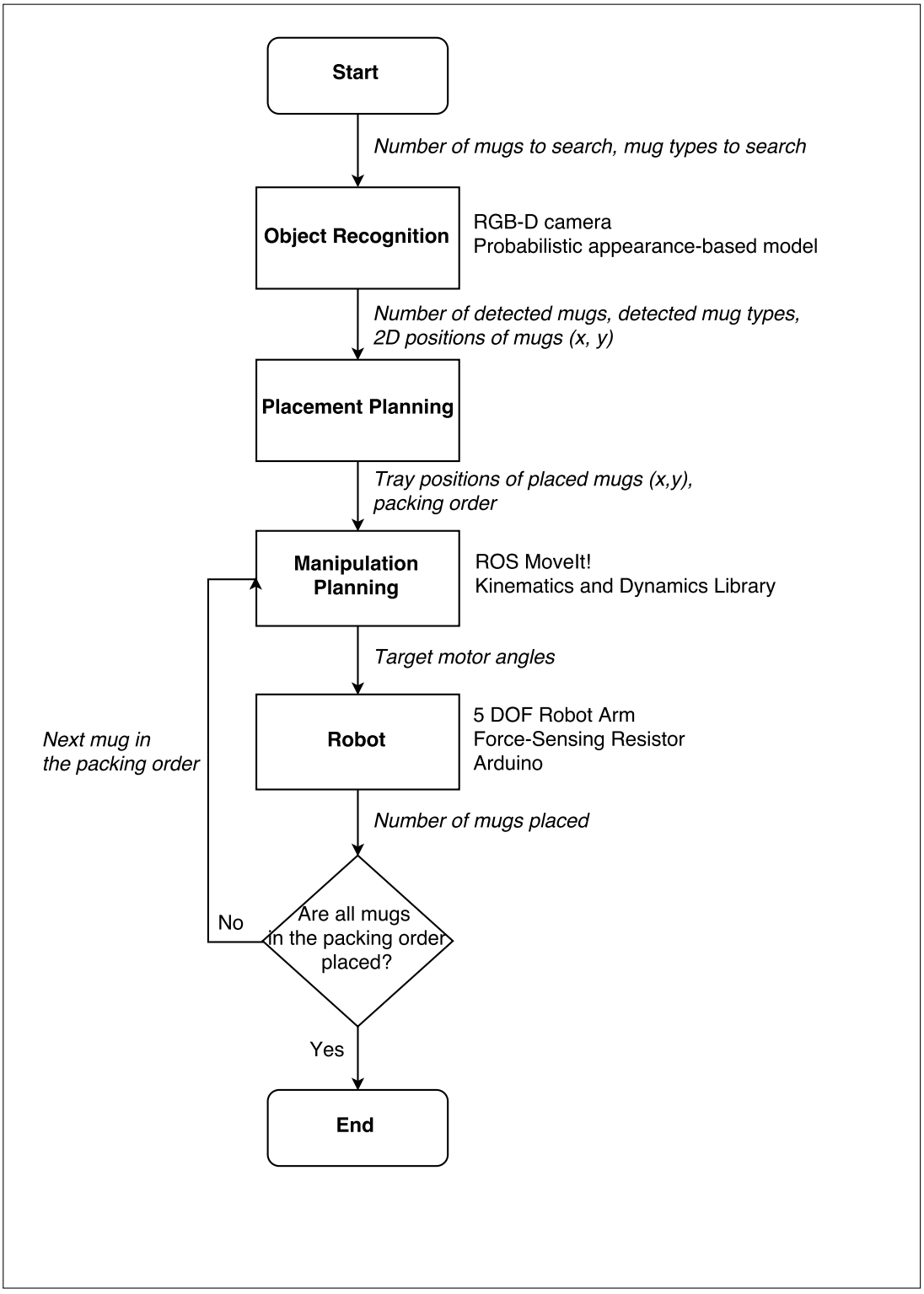


Figure 3.6. The processing flowchart of our system.

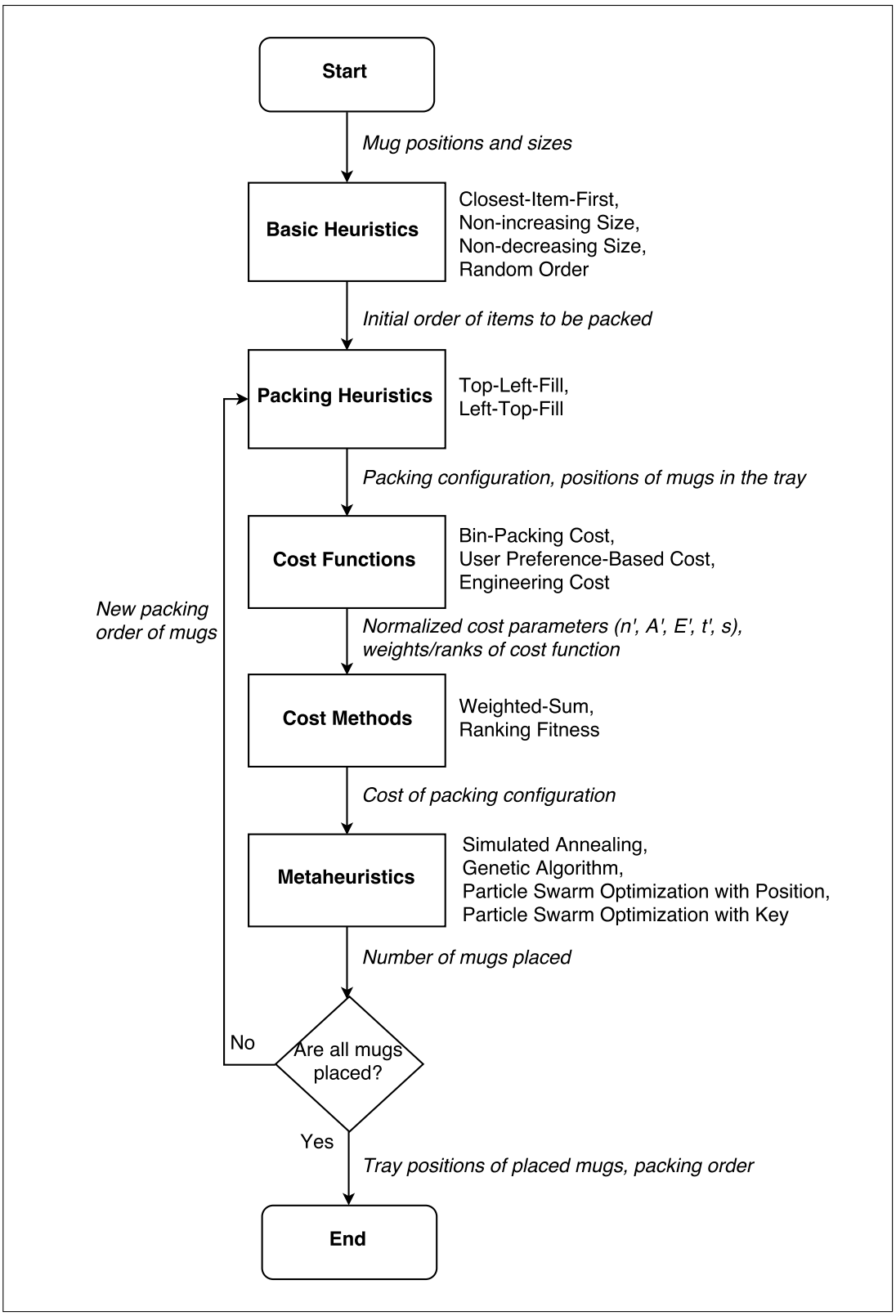


Figure 3.7. The flowchart of placement planning.

4. METHODOLOGY

The execution of our system consists of three parts: object recognition, placement planning, and manipulation planning. First, the mugs on the counter-top are recognized according to their types, and their pose estimates are made, which is described in Section 4.1. Afterward, according to the number of mugs and the sizes of the mugs and the tray, a placement within the tray is found using placement planning, as described in Section 4.2. Three cost functions with two different application methods are used in planning the mugs to place into the tray, which are described in Section 4.2.1. Finally, all mugs that are planned to be placed on the tray are placed using manipulation planning, as explained in Section 4.3.

4.1. Object Recognition

We collaborated with Özgür Erkent for the object detection algorithm, and used their code in our project [90]. They use probabilistic appearance-based model, in their approach, for the object detection and pose estimation framework. The objects are learned through the training dataset by extracting edge points and orientation, depth range, hue color and surface normals from a given image. Each of these features is defined by its position in the image and its appearance attributes. The appearance attributes contain local orientation of an edge point, depth values of the object, hue values, and the surface normals. A Gaussian Kernel distribution is obtained from the training set, where each image corresponds to one viewpoint. Another distribution is derived from the test set images, in which the viewpoint is unknown. The similarity measure between the test and the training images is obtained through the cross-correlation of the training and test distributions. The samples are drawn by Monte Carlo methods, in order to increase efficiency in their method. The objects that have a similarity score above a certain confidence level are accepted. 6D poses, i.e. the 3D position coordinates and 3D orientation coordinates, of the center of each recognized object is found through their algorithm.

We conducted intrinsic calibration for the *ASUS Xtion Pro Live* camera [101]. We provided the training dataset of our mugs through 3D PLY models of the objects that we

created through SolidWorks [102]. The models are shown in Figure 4.1. We are using two types of mugs, which are explained in detail in Chapter 5. The dataset for testing purposes is obtained through RGB and depth data images by the *rosbag* and *openni_launch* modules of *ROS* [103]. The algorithm is tested in real-time data from the depth camera, during the execution of the program.

(a) (b)

Figure 4.1. Mug models: (a) Mug type 1 (big mug), (b) Mug type 2 (small mug).

Object recognition is conducted at the beginning of our algorithm. After the mug set on the counter-top is found, placement planning is conducted in order to optimize the planning inside the tray. Then manipulation planning is used to place the mugs onto the tray accordingly to the plan of the placement. However, during any stage of the manipulation, when object detection is necessary, the object detection algorithm is run again and the currently detected mugs are compared to the previously detected set. If the detected set has less or greater number of mugs than the previously detected set, placement planning is conducted to find a new placement with the remaining number of mugs. Otherwise, a tolerance amount of 1 cm in any direction is given around the previously detected mugs, in order to compare them to the current detected positions. If a current detected mug is in that range of the previously detected mug, that mug is labeled as "known", and its position information does not change. If at the end of the comparison of the previous and the current detected mug set, there is only one "unknown" mug, then that mug is the current mug to be placed. Its position information is updated and then the manipulation planning continues to place that mug to the tray. If there are multiple "unknown" mugs, then the current mug to be placed is found from the mug type since the mug to be placed currently has a determined type according to the placement planning. If there are multiple "unknown" mugs of the same type as the

target mug, the “closest” mug to the robot is chosen as the current mug, in order to minimize the energy spent in the manipulation. For the rest of the “unknown” mugs, the assignment to the previous mug set is made by their closeness to the previous positions of the mugs. Unassigned mugs and their position information are updated. The flowchart of our approach is presented in Figure 4.2.

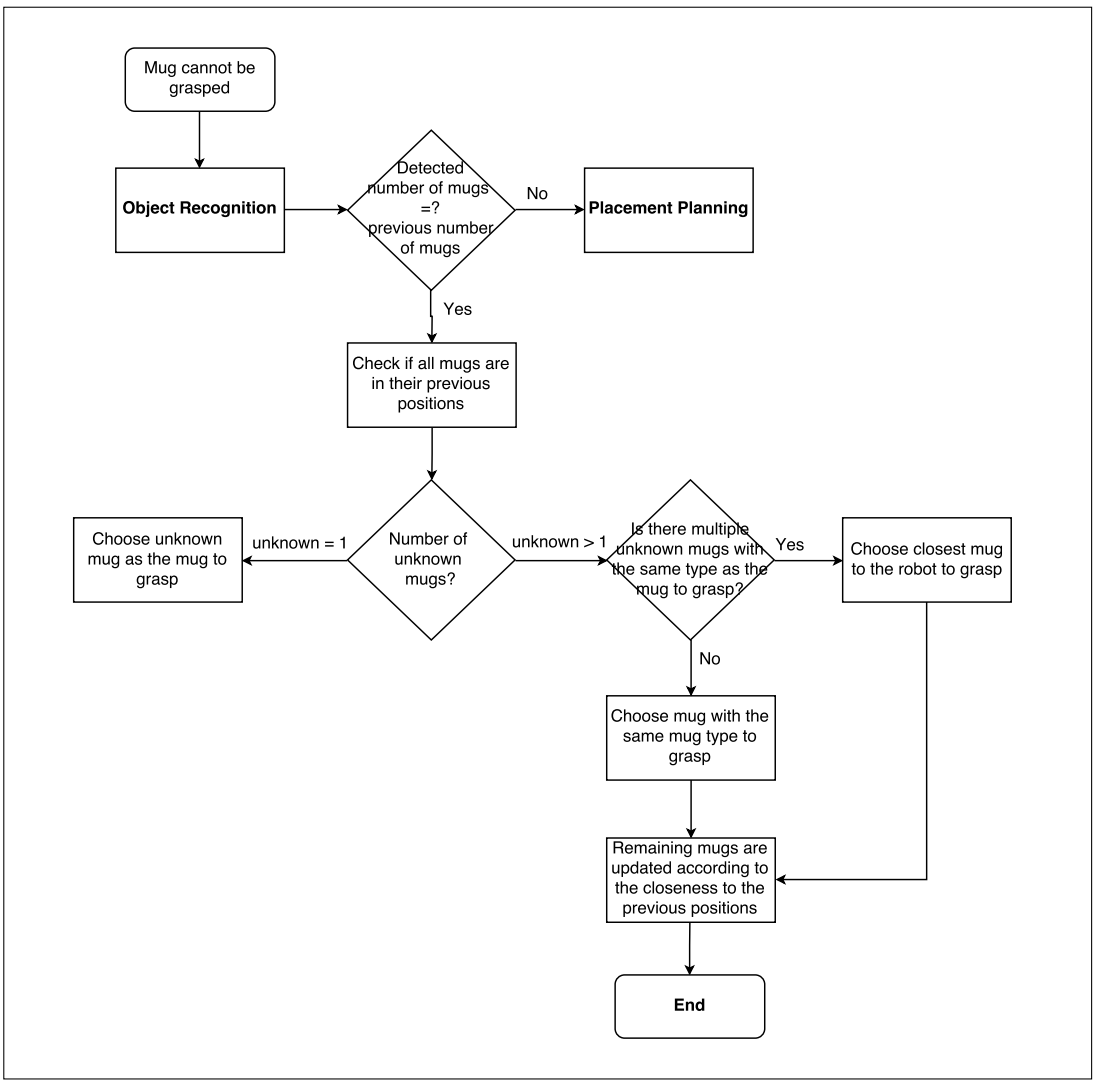


Figure 4.2. Processing flow of finding a lost mug.

The downside of this object recognition algorithm is that it requires to specify how many mugs there are on the counter-top, along how many of each type of mug exists. As a solution, a fixed number of mugs can be specified beforehand that would exceed the actual number of mugs on the counter-top, and the algorithm will find as many unique items as

possible. However, this increases the time required for the algorithm to find the solutions.

4.2. Placement Planning

We aim to optimize the placement for the mugs within the tray in our project. Since the mugs will not be stacked on top of each other, in order to allow them to be cleaned properly in the dishwasher, we can simplify our 3D packing problem to a 2D packing problem. As stated before, 2D packing problems are proved to be NP-hard [24], hence, we use methods that result in a near-optimal solution. We compare three methods for this problem, which are used in this area as mentioned in Section 2.1: SA, GA, and PSO. We use all three methods in order to find the best method for finding the near-optimal solution for the placement of mugs into the dishwasher. These methods require a fitness function to compare neighboring solutions, therefore, we decided to find a cost function which would satisfy the needs of our project and the people, which is described in Section 4.2.1.

We use rectangular bounding boxes for mugs, and the sizes of the mugs are defined from the top view of the mug since the mugs will be placed upside down on the tray. The "length" (l_m) of the mug is defined as the diameter of the brim and the "width" (w_m) is defined as the size including the mug handle facing either left or right, and the brim of the mug, as in Figure 4.3. The mugs are brought to that position before they are placed into the tray by the manipulator. The tray is modeled as an empty 2D box, with the width defined in horizontal direction and length defined in the vertical direction with respect to the view of the robot. All the mugs positions are perceived through object recognition before the calculation of the placement within the tray. Any random value used in the algorithms is chosen from a uniform random distribution.

4.2.1. Cost Function

We analyzed the packing problem as an engineering problem, and we found that the following criteria have to be optimized in the case that a robot places the dishes:

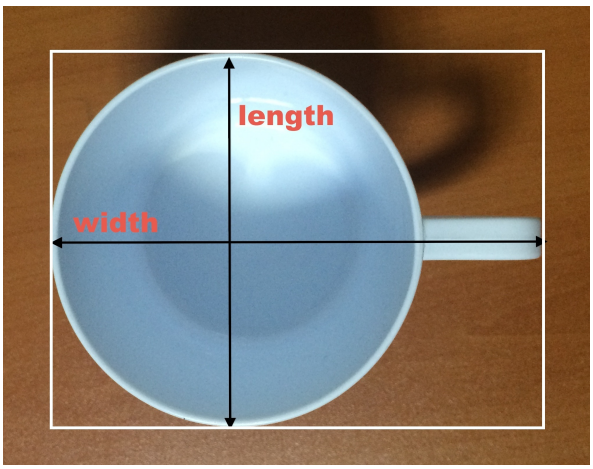


Figure 4.3. Mug length and width.

- Maximize number of mugs placed
- Minimize energy to place the mugs
- Minimize time required to solve the problem
  - (i) Minimize time for solving the problem
  - (ii) Minimize time for placing the mugs

However, we intend our robot to work in a house environment, which required us to find the criteria that a person would consider if a robot was placing their own dishes. Therefore, we made several studies in order to learn these criteria and find their weights in our multi-objective cost functions.

4.2.1.1. Preliminary and User Studies. A preliminary study was made in order to find the criteria that people consider when they are placing dishes into the dishwasher. In this study, we asked our subjects in person an open-ended question, ”What criteria would be important to you if you had a robot that would place the dishes into the dishwasher?”. The results of this study are presented in Section 5.2.1. Through this study we found that there is an additional parameter that we have to consider for the packing problem: “sortedness”, that is, placing the dishes according to dish size, such that the bigger dishes are separated from the smaller ones.

The multi-objective packing problem requires a method for comparing the fitness values for two solutions. Therefore, we decided to use two different fitness methods (f_m) for comparing the solutions: the weighted normalized sum of parameters presented above, and the ranking method, in which the parameters are compared with a prioritized order. However, both of these methods required an ordering of these parameters. For this reason, we conducted an anonymous online survey, in which we asked our subjects to rank these parameters in the order of importance if they had a robot placing their dishes. We also provided the option of "not important" for the parameters. In the questionnaire, we also requested their gender, age range, working status, if they are currently a student, if they have a dishwasher, how often they use their dishwasher and approximately how many minutes would it take for them to place dirty mugs into the dishwasher along with the ranking question. The survey was made in Turkish. The survey is presented in Appendix B and the weights found in the study according to different categories are presented in Section 5.2.2.

4.2.1.2. Observational Study. In order to find the maximum possible placements for our datasets, we decided to conduct an observational study. Prior to the experiments in the study, we asked our participants to sign a participant information and consent form shown in Appendix C, which is formatted accordingly to the information and consent form by Bogazici University ethics committee [104]. Moreover, we requested them to fill the questionnaire, that we conducted before, with three additional questions. Two of these questions depended on the possession of a dishwasher: whether they load their dishwasher, and whether they unload their dishwasher, in separate questions. These questions allowed us to examine the categories for the cost function in more detail. We also asked why is the sortedness parameter important for them if they marked the sortedness as important, for understanding the reasoning behind this parameter. The questionnaire is presented in Appendix C.

In the experiments, we asked our users to place mugs on the counter-top to the given tray. We had two types of trays: the smaller tray, which we use in our robot experiments, and a bigger tray that have equal sizes to a real dishwasher tray. We had six datasets, three for each tray, with a different number of mug types. The datasets and the tray and mug sizes are given in Chapter 5. During their placement, we requested them to think aloud. We

noted down their train of thoughts and examined their packing method. For each dataset, we counted the number of mugs placed, measured the placement time, and took a photo of their placement. Each experiment was 30 to 45 minutes long.

After the experiments, we asked our participants to fill an Obsessive-Compulsive Scale test adapted from [105], which is an adapted version of [106]. This test is not a clinical test. It is to help us determine whether the participant might have an obsessive-compulsive disorder (OCD) and our participants' names are not revealed in this study, and they will not be revealed in the future. The result categories for the test are "unlikely", "probable", and "likely", in increasing OCD probability, and they are found through scaling the thresholds for the OCD test used in [105]. Our objective behind using this test is understanding the reasoning behind the importance of the sortedness parameter. People who tend to have OCD are more prone to being concerned with the symmetry of the items, which we thought could be the reason behind at least some of the answers for the importance of the sortedness parameter. The Obsessive-Compulsive Scale test is given in Appendix C. The results of these experiments and the questionnaires are presented in Section 5.2.3.

4.2.1.3. Annotation. In order to analyze the sortedness of each of the placements of the participants in the observational study, we decided to have them annotated by other people. Therefore, we made an annotation website in PHP, in which a participant can signup or login with a valid e-mail and password, that he/she creates the first time he/she use to the system. This allows the participant to login at any time and also retrieve the password if he/she forgets it through e-mail. The e-mails of the participants are undisclosed in this study, and their passwords are hashed in the database with *BCrypt* [107] in PHP. The participants are asked to fill the first questionnaire from the experiments, with the additional question of which is the maximum amount of time you would want for your robot to fill your dishwasher. For the annotation phase, the participants are asked to rate the sortedness with respect to the mug sizes of the placement (image) with a Likert scale, with values "very ordered","ordered", "moderately-ordered", "unordered", "very unordered". The first three images a participant rates are artificial placements for the hardest dataset, i.e. dataset with the highest amount of big mugs for the bigger tray (dataset 4), which is intended to correspond to "very ordered",

"very unordered" and "average". Hence, we train our participants with these three images, without biasing them, since the measure of sortedness depends on the eye of the beholder. These images are followed by a random sequence of 42 images of placements for the two datasets. The datasets to be rated for each participant are determined by the number of previous annotators. Modulus of 3 is applied to the annotator number. If the result is 0, then the participant rates the placements from datasets 3 and 6, and if the result is 1, he/she rates datasets 1 and 4, and finally, if the result is 2, he/she rates datasets 2 and 5, such that all the placements are rated if at least 3 people have participated in the annotation. Due to the high number of images to be annotated, we decided to use animated GIF images per 7 annotation images. This approach allows the participant to rest at least 5 seconds, which is the duration of the GIF files, such that they can regain their concentration. Following the annotation phase, the participant is requested to fill the OCD test as described before. The results of the annotation and the questionnaires are presented in Section 5.2.4. The annotation pages and the questionnaires can be found in the Appendix D. The annotation website is available online [108].

4.2.1.4. Parameter Definitions. We use cost functions as the fitness functions in our algorithms and our aim is to minimize the cost. In order to eliminate any scale differences in our parameters, we used feature scaling, i.e. unity-based normalization method, for normalizing our parameters to [0,1] range. The sortedness (s) parameter is used for placing the dishes according to dish size: it is 0, if an optimization algorithm, i.e. SA, PSO or GA, found a solution that has, at least, one mug which is not in order according to the size. Hence, each solution is compared with non-increasing width, non-increasing length, non-decreasing width, non-decreasing length methods. If according to any of the sorted-size methods, the solution has an ordered solution, then sortedness is 1, because the mugs are sorted according to their sizes in those methods, otherwise it is 0. The sorted-size methods have sortedness as 1. No normalization is necessary for this parameter since it can be only 0 or 1.

We use potential energy for estimating the electrical energy spent by the motors as in

Eq. 4.2.

$$\Delta m = \sqrt{(x_{mtr} - x_{mc})^2 + (y_{mtr} - y_{mc})^2} \tag{4.1}$$

where Δm is the Euclidean distance traversed in order to place a mug, that is, to move the mug from its position on the counter-top (x_{mc}, y_{mc}) to the finally to the mug position on the tray (x_{mtr}, y_{mtr}). The distance traveled is an underestimate of the total distance, therefore the energy used in placing a mug is found to be less than the real energy. However, the energy is normalized in our approach, therefore, the real energy is not required for comparing two solutions.

$$\begin{aligned} F &= M_m G \\ E &= \sum_{1}^{n} F \Delta m \\ E' &= \frac{E - E_{min}}{E_{max} - E_{min}} \end{aligned} \tag{4.2}$$

E is the electrical energy, E' is the normalized energy, n is the number of placed mugs, F is the force exerted by the arm, M_m is the mass of a mug, and G is the gravitational acceleration.

Similarly, normalized number of placed mugs (n') is found by Eq. 4.3.

$$n' = \frac{n - n_{min}}{n_{max} - n_{min}} \tag{4.3}$$

The covered area of the tray by the mugs is calculated as below:

$$\begin{aligned} A &= \sum_{1}^{n} (w_m + \Delta_w)(l_m + \Delta_l) \\ A' &= \frac{A - A_{min}}{A_{max} - A_{min}} \end{aligned} \tag{4.4}$$

where A is the covered tray area, w_m is the mug width, l_m is the mug length and Δ_w, Δ_l are the tolerances between mugs in width and length. The placement time (t) is calculated as

follows:

$$t = t_c + \frac{\sum_1^n \Delta m}{V_a}$$
$$t' = \frac{t - t_{min}}{t_{max} - t_{min}} \tag{4.5}$$

where t' is the normalized placement time, t_c is the placement calculation time and V_a is the speed of the arm. t_c is calculated as the calculation time of each iteration. The overall calculation time is not considered for comparing the neighboring solutions, because this would result in the solutions that are found later in the iterations to be discarded, which would be undesirable.

The maximum for each of the parameters is updated at runtime when a greater parameter is calculated within the iterations of the optimization algorithms (SA, PSO or GA). The minimum is updated in the same way. We keep the values of the parameters that correspond to the minimum cost such that if an update occurs for the maximum values, the previous minimum cost can be recalculated. The calculated cost(s) is compared to the updated previous minimum cost(s). If the value is smaller than the minimum cost, the minimum cost is updated, along with the parameter values at that configuration.

4.2.1.5. Fitness Methods. We use two types of methods for comparing the cost of two solutions: weighted sum and ranking method. The cost for the weighted sum of the parameters method is calculated as follows:

$$C(n,E,t,s) = W_n(1-n') + W_E E' + W_t t' + W_s(1-s) \tag{4.6}$$

where W_n, W_E, W_t, W_s represent weights for the number of placed mugs, energy, time and sortedness, respectively, which are found through the user studies. In order to maximize number of mugs placed, and maximize the sortedness, each of the normalized values is subtracted from 1.

We use Eq. 4.7 for finding the weights (W) for the cost function.

$$W_i = \frac{\sum_{j=1}^{N} R_{ij}(N-j+1)}{\sum_{i=1}^{N}\sum_{j=1}^{N} R_{ij}(N-j+1)} \tag{4.7}$$

where i, j, N, R_{ij} represent parameter number, rank, number of parameters, number of responses for a parameter i with rank j, respectively. If a parameter is "not important", the rank is assumed to be zero, hence it is not included in the calculation for the weights of the cost function.

For the ranking of the parameters method, the cost of two solutions is compared by comparing each parameter in an order. The value of the parameter with the highest ranking, i.e. highest weight, for the current solution is compared with the best placement, which gives the lowest cost. If the parameter has a lower value than the value for the best placement, then the cost is said to be lower. Otherwise, if they are equal, the second parameter in the ranking order is compared with the second parameter of the best case, and so on. The normalized covered area (A') is considered in this method in order to use the maximum capacity of the dishwasher. If two placements have the same number of mugs placed, then the solution with a higher covered area is preferred. In order to have the same structure as the weighted sum cost, each of the normalized values of the number of mugs, covered area, and sortedness is subtracted from 1. The ranking order of parameters are found through the weights of the weighted-sum method. The parameters are sorted according to non-increasing weights, and the corresponding fitness ranks are found from the sorted order of the parameters.

4.2.1.6. Types of Cost Functions. We use three types of cost functions (f_f) for each of the fitness methods:

- The bin-packing cost (C_b), i.e. using the maximum capacity of the dishwasher
  (i) The number of mugs is used for the weighted sum method
  (ii) The number of mugs and the covered area are used for the ranking method

- The user preference-based cost (C_u), i.e. the overall cost for all the parameters
- The engineering cost (C_e), i.e. the cost without sortedness

The bin-packing cost function (BPC) is the general fitness function used in the bin-packing problems. As mentioned in Section 2.1, the number of packed items, the covered area, or the volume utilization is considered for this function in the literature. We consider only the number of mugs for the weighted-sum method, such that the maximum number of mugs can be placed. On the other hand, we also consider the covered area of the tray in the ranking fitness method, such that a placement that maximizes both of these criteria can be found. The covered area cannot be used in the bin-packing cost function for the weighted sum method because the weights of these parameters cannot be determined accurately, as they are both important. However, maximizing the number of mugs placed is more important, because it would reduce the remaining number of mugs on the counter-top to place for the next time the dishwasher is loaded. The user preference-based cost function aims to optimize all of the parameters defined in Section 4.2.1.4, based on the preferences of people. The engineering cost function (EC) aims to optimize the criteria in the engineering aspect of this problem, as we considered initially, therefore, sortedness parameter is omitted in this function.

For the weighted sum method, the equations of cost functions are presented in Eq. 4.8 for the bin-packing cost, Eq. 4.9 for the user preference-based cost, and Eq. 4.10 for the engineering cost functions.

$$C_{b_{WS}} = (1 - n') \tag{4.8}$$

$$C_{u_{WS}} = W_n(1 - n') + W_E E' + W_t t' + W_s(1 - s) \tag{4.9}$$

$$C_{e_{WS}} = W_n(1 - n') + W_E E' + W_t t' \tag{4.10}$$

The bin-packing and the engineering cost functions are specialized versions of the user-preference based cost function, where the weights for the parameters that are not used are zero. For the ranking fitness method, the parameters are ranked in order of importance as described in Section 4.2.1.5. The bin-packing cost function for the ranking fitness method is given in 4.11.

$$C_{b_{RF}} = \begin{cases} 0 & \text{if } n'_c > n'_p \text{ or } \left(n'_c = n'_p \text{ and } A'_c > A'_p\right) \\ 1 & \text{otherwise} \end{cases} \tag{4.11}$$

where n'_c, A'_c denote the number of mugs placed and the covered area for the current placement, and n'_p, A'_p denote the number of mugs placed and the covered area for the previous placement. Likewise, the other cost functions are calculated by ranking the parameters in the order of their importance. For the user preference-based method n', A', E', t', s are compared, whereas, for the engineering-cost, sortedness (s) is omitted from the comparison.

In order to compare the neighboring solutions for all the placement methods, i.e. SA, GA, and PSO methods, the cost parameters are stored for each iteration τ. Therefore, for each method, the total cost (C_τ), the cost parameters $((1-n'), (1-A'), t', E', (1-s))$ denoted as $\vec{C}_\tau$, the order of the placed items according to the packing heuristic (o_τ), the vector of placement parameters (n, A, t_c, t, E, s) denoted as ϕ_τ are used. Hence, these fitness parameters are denoted as Ψ_τ, for the ease of use, throughout this study.

4.2.2. Packing Heuristics

For the packing method, we use two types of heuristics (h): Bottom-Left-Fill and Left-Bottom-Fill heuristics. In the Left-Bottom-Fill (LBF) heuristic, we use the same approach as in BLF but the mugs are placed at the most left location, as close to the bottom as possible, instead. Because of the lack of a wrist yaw joint, the mugs cannot be rotated in the Z axis, hence, the placed mug handles are in the X direction or are slightly rotated from that direction since the arm is rotated towards the target position.

The initial input list of items to be packed through BLF or LBF for our optimization methods are determined through one of these methods, which represent the sort method (κ):

- Closest-item-first (CIF): The items are sorted in the X axis, than in the Y axis for this ordering. The items that are closer to the robot in the x coordinate, are at the beginning of the list. If two x coordinates are the same, the item with the y coordinate closer to the robot comes prior than the other.
- Non-increasing size (NIS): The items are sorted in non-increasing order according to their size.
  (i) Non-increasing length: The items are sorted in non-increasing order according to their length.
  (ii) Non-increasing width: The items are sorted in non-increasing order according to their width.
- Non-decreasing size (NDS): The items are sorted in non-decreasing order according to their size.
  (i) Non-decreasing length: The items are sorted in non-decreasing order according to their length.
  (ii) Non-decreasing width: The items are sorted in non-decreasing order according to their width.
- Random-ordering (RO): The order of the items are randomized.

Each of these orderings is separately tested with our optimization methods, in order to observe the effect of the initial placement to the optimization process.

We use an occupancy matrix (om) to determine if a current cell is occupied by a mug or not. The matrix has a cell size of 0.1 cm $\times$ 0.1 cm, because the object detection has a resolution of millimetric scale. In case a smaller size than the actual size of mug is detected or if the mug is rotated slightly during the placement process, a tolerance amount, both in width (Δ_w) and length (Δ_l), is used between the mugs to compensate for those effects. The tolerance is the empty space between two mugs in the horizontal or vertical direction. The number of rows (R_{om}) and the number of columns (C_{om}) of the occupancy matrix are found

by

$$R_{om} = \frac{l_{tr}}{l_{cl}} \tag{4.12}$$

$$C_{om} = \frac{w_{tr}}{w_{cl}} \tag{4.13}$$

where l, w, tr and cl corresponds to length, width, tray and cell, respectively. A placed mug occupies,

$$R_o = \left\lceil \frac{l_m + \Delta_l}{l_{cl}} \right\rceil \tag{4.14}$$

number of rows and

$$C_o = \left\lceil \frac{w_m + \Delta_w}{w_{cl}} \right\rceil \tag{4.15}$$

number of columns in the occupancy matrix, where m denotes mug. In order to determine whether a mug can fit in a position or not, that many number of rows and columns are checked in the occupancy matrix.

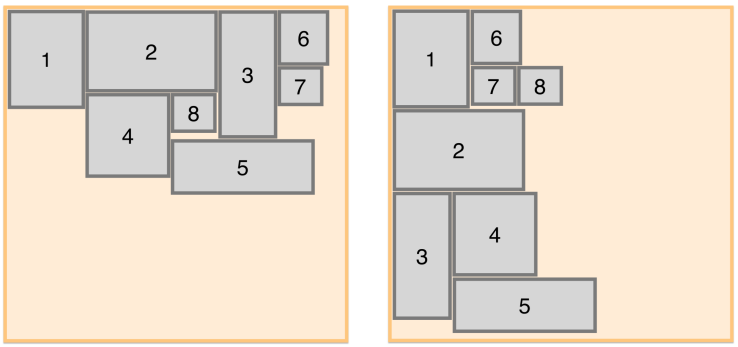
Since during the experiments the BLF or LBF heuristics caused arm collisions with previously placed mugs at lower levels, we have used "Top-Left-Fill" (TLF) or "Left-Top-Fill" (LTF) heuristics, in which mugs were placed on the top-most level, instead of the bottom-most level. Figure 4.4 shows the collision between the wrist and the previously placed mug. The TLF and LTF heuristics are shown in Figure 4.5.

4.2.3. Simulated Annealing

Simulated Annealing (SA) [25] algorithm is an iterative improvement metaheuristic. In the algorithm, the "temperature", "absolute temperature" and "cooling rate" are used for determining the number of iterations and "energy" is used for representing the objective

(a) (b)

Figure 4.4. (a) Gripper collision with a previously placed mug in the Bottom-Left-Fill approach, (b) No collisions in "Top-Left-Fill" approach.



(a) (b)

Figure 4.5. (a) Top-Left-Fill, and (b) Left-Top-Fill heuristic.

function. At the beginning of the iteration, the "temperature" starts with a defined value, and it decreases with a "cooling rate" as the iteration continues. The algorithm is initialized with a known initial configuration and a neighboring configuration is created by altering a given state in some well-defined ways and that configuration is used as the new configuration if it is acceptable by the acceptance probability function, in order to avoid a local minimum. The acceptance probability function is defined by:

$$P(e,e',T) = \begin{cases} 1 & \text{if } e' < e \\ \exp((e-e')/T) & \text{otherwise} \end{cases} \tag{4.16}$$

where e is the "energy" of the current configuration, e' is the "energy" of the neighboring configuration and T is the current "temperature". The neighboring configuration is accepted as the new configuration if it is better than the previous configuration in terms of the optimization parameters ($e' < e$) or if the exponential of the "energy" difference over the current "temperature" T is greater than a random value between 0 and 1. At lower "temperatures" $P(e, e', T)$ will be lower hence the probability of accepting a worse solution will be less. The stopping criterion is reached when the "temperature" is below some absolute "temperature" or if a heuristic is satisfied, in the case of hybrid algorithms. This algorithm does not guarantee a global optimum, however, it decreases the time required for a complete search of the solution space. The pseudocode for the Simulated Annealing algorithm used in our study is given in Figure 4.6, where n represents the number of mugs in the placement, cr is the cooling rate, S is the configuration, and Λ is the placement order set of the mugs.

In our approach, the "energy" parameter is the cost of the fitness function and it is calculated depending on the fitness method as described in Section 4.2.1. The solution is accepted if the cost is lower or if the acceptance probability function is greater than a uniformly distributed random real number in the range [0,1]. The "energy" for the ranking method is found by summing the costs of each parameter, and dividing them by the number of parameters, in order to allow for the acceptance probability function to be calculated in the case that the cost is higher. The initial configuration (placement) is created through either TLF or LTF methods for the list of items to be packed, as described in 4.2.2.

In order to create a neighboring configuration, we swap the ordering of two mugs for SA, and rearrange the mugs within the tray with that new order, according to either TLF or LTF heuristic. A placed mug can be swapped with a placed or not-placed mug, as long as these two mugs do not have the same width and length. After the rearrangement, we check if a new mug can be placed from the not-placed set. We sort the mugs according to the non-decreasing area, and we select the mug with the minimum area from the not-placed set and check if we can add that mug to our current placement. If it can be added, we remove it from the not-placed set and we look at the next mug with the minimum area in the not-placed set until a mug in that set cannot be placed, or until all the mugs in the not-placed set are placed. We change the configuration to the new configuration with the extra added mug(s), if there

Input $h, \kappa, f_m, f_f, n_{total}, T_{max}, T_{abs}, cr, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$

Output S_{best}

$S_{cur} \Leftarrow$ CreateInitialPlacement($h, \kappa, f_m, f_f, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$)

$S_{best} \Leftarrow S_{cur}$

$T \Leftarrow T_{max}$

while $T > T_{abs}$ **do**

  $S_{new} \Leftarrow$ CreateNeighborSolution($h, f_m, f_f, S_{cur}, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}$)

  Calculate $e(S_{new})$

  if $e(S_{new}) \leq e(S_{cur})$ **then**

    $S_{cur} \Leftarrow S_{new}$

    if $e(S_{new}) \leq e(S_{best})$ **then**

      $S_{best} \Leftarrow S_{new}$

    end if

  else if $\exp(-(e(S_{new}) - e(S_{cur}))/T) > \text{rand}(0,1)$ **then**

    $S_{cur} \Leftarrow S_{new}$

  end if

  if $n(S_{cur}) = n$ **then**

    Break

  end if

  $T \Leftarrow$ DecreaseTemperature(T, cr)

end while

Return(S_{best})

Figure 4.6. Pseudocode for Simulated Annealing Algorithm.

is any. Afterward, the acceptance probability function is checked, and if it is accepted, the configuration is updated. The placement with the maximum number of mugs placed is kept, and the final placement configuration is changed with the maximum if the resulting number of placed mugs is less than the maximum. If all the mugs on the counter-top are placed at a configuration then the iteration stops, otherwise, the iteration continues until the temperature is below the absolute temperature. Since the mugs have varying sizes and the problem is NP-hard, the optimum number of mugs that can be placed in the tray cannot be determined or approximated beforehand, therefore, the stopping criterion does not include a mug limit that the algorithm should satisfy. Instead, the algorithm randomly searches the search-space by swaps, hence, it will not always give the same number of mugs. Increasing the number of iteration cycles may result in a higher number of mugs because a bigger search-space will be scanned with swaps, however, it will also increase the calculation time for the algorithm. The temperature and cooling rate parameters were adjusted empirically in order to find a high amount of mugs and a reasonable amount of time for the algorithm. The pseudocode for creating a new neighbor solution is given in Figure 4.7.

4.2.4. Genetic Algorithm

Genetic Algorithm [40] is a metaheuristic search method, that imitates the generation of a population. Each solution in the search-space is represented by a "chromosome", and a set of solutions is called a "population". The population created at an iteration is called the "generation". A "chromosome" consist of "genes", analogous to genetics.

In our approach, we combine several methods from the literature. A "gene" (ς) is encoded as $(\imath, w, l, \vec{x})$, where the parameters denote the number of the item in the original packing order, width, length, and the position vector, respectively. A "chromosome" (λ) is represented with $\vec{\varsigma}$ and Ψ_λ parameters. A "population" (ρ) is a vector of "chromosomes", which are also called "individuals" in the "population". A "generation" is a population at iteration τ.

In our problem, the number of total items might be greater than the number of items that can be packed. Therefore, in order to use a fixed-size "chromosome", we adopt the

Input $h, S_{in}, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}$

Output S_{new}

$i \Leftarrow \text{rand}(n_{total})$

$j \Leftarrow \text{rand}(n_{total}), j \neq i$

while $size(S_{in}(i)) = size(S_{in}(j))$ **do**

  $j \Leftarrow \text{rand}(n_{total}), j \neq i$

end while

$\Lambda \Leftarrow \text{GetPlaceOrderOfConfiguration}(h, S_{in})$

$\Lambda \Leftarrow \text{Swap}(\Lambda(i), \Lambda(j))$

$S_{cur} \Leftarrow \text{Rearrange}(h, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda)$

while $n(S_{cur}) < n_{total}$ **do**

  $m \Leftarrow \text{FindNotPlacedMugWithMinArea}$

  $p(n(S_{cur}) + 1) \Leftarrow m$

  $S_{new} \Leftarrow \text{Rearrange}(h, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda)$

  if $n(S_{new}) > n(S_{cur})$ **then**

    $n(S_{cur}) \Leftarrow n(S_{cur}) + 1$

    $S_{cur} \Leftarrow S_{new}$

  else

    Break

  end if

end while

$S_{new} \Leftarrow S_{cur}$

Return(S_{new})

Figure 4.7. Pseudocode for CreateNeighborSolution function.

structure of Gehring and Bortfeldt [33] and Hopper and Turton [19]. We use a "chromosome" size that is equal to the number of total items as if we are using a virtual container. The population is sorted after the creation of each "chromosome", such that, those with the lowest costs come prior than others. The initial population is created by one "individual" who has a sorted-order according to the sort method (κ), and the rest of the population is created with random permutations of items, as described in [42, 46]. No duplicates are allowed in the population. Search is aborted when a placement with all of the items are found. We adopted the method by Gehring and Bortfeldt [43, 44] for the selection of a "parent" in crossover or mutation operations. The first "parent" is selected to be the first "individual" in the sorted population, that is, the "individual" that has the lowest cost. The second "parent" for the crossover is chosen randomly. We use Partially Mapped crossover (PMX), as used in [19], and the mutation operation is an inversion, that is swapping two items positions in the ordering, as used by [33, 45]. We use the placement packing methods described in Section 4.2.2. The cost functions which are used for comparing two "chromosomes" are as given in Section 4.2.1.

A new "generation" is created as follows:

- Elitist strategy: $nrep$ best "individuals" from the previous "generation" are reproduced
- $npop - nrep$ number of "individuals" are created through crossover with PC probability, and through mutation with PM probability
  - (i) Each "individual" ordering is placed within the tray through the packing method (h)
  - (ii) The "chromosome" of the "individual" is reordered according to the items placed
  - (iii) If the "chromosome" is not a duplicate "chromosome" in the "population", the fitness values of the "individual" are calculated
  - (iv) If a placement with a higher or lower placement parameter is achieved, the previously calculated fitness values of the "population" are updated
  - (v) "Population" is sorted according to the fitness values

The procedure of our approach is given in Figure 4.8. In the procedure, $ngen$ is the number of "generations" to be created, $npop$ is the size of the population to be created, gs

denotes the current "generation" ("population") size, *os* is the offspring from the crossover or mutation operation, $C_{\lambda_{os}}, \vec{C}_{\lambda_{os}}$ are the cost and the cost function parameters, λ_b is the best "individual" ("chromosome") in a generation.

4.2.5. Particle Swarm Optimization

Particle Swarm Optimization [47] is a metaheuristic method, which is used for the optimization of a problem. It solves a problem by having a population of candidate solutions, which are called "particles". These "particles" are moved around in the search-space with a "velocity". The "velocity" of a "particle" depends on the previous "velocity", best-known position of the "particle" and the global best "particle" that has the best solution in the overall. The idea of using a global best solution is to have the "particles" move towards the best solution, to optimize the problem.

We use two methods for encoding the 2D bin-packing problem to a "particle":

- A "particle" is represented with the (x, y) position of the item in the tray
- A "particle" is represented with the list of all items sorted according to their keys

The first method is denoted as PSOp, and the second method is denoted as PSOk throughout this study. PSOp variables that are different in representation or calculation method are denoted by the superscript ^{p} from PSOk variables, which have ^{k} as the superscript.

4.2.5.1. PSOp. We modified the PSO method proposed by Shin et al [51, 52]. Our modifications to their approach are as follows:

- "Particle" (ρ_i^p) representation and the introduced "particle ordering" structure (χ_τ^p)
- Lower ($\vec{B}_{L_i}$) and upper bound ($\vec{B}_{U_i}$) calculations
- Initialization of position ($\vec{x}_i^p$) and "velocity" ($\vec{v}_i^p$) of particles
- Integration of the fitness functions and the fitness methods for the comparison of two

Input $h, \kappa, f_m, f_f, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$

Output λ_b

$\phi_{\lambda_{min}}, \phi_{\lambda_{max}} \Leftarrow 0$

CreateInitialGeneration($h, \kappa, f_m, f_f, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \phi_{\lambda_{min}}, \phi_{\lambda_{max}}, \Lambda$)

for $\tau \Leftarrow 1$ **to** $ngen$ **do**

  for $i \Leftarrow 1$ **to** $nrep$ **do**

    $\rho_\tau(i) \Leftarrow \rho_{\tau-1}(i)$

  end for

  while $gs < npop$ **do**

    Pick random r with discrete distribution with probabilities $[PC, PM]$

    $\lambda_b \Leftarrow \rho_\tau(0)$, $i \Leftarrow \lambda_b$

    if $r =$ crossover operation **then**

      $j \Leftarrow$random$[1, gs-1]$, $\lambda_{os} \Leftarrow$ Crossover(n_{total}, i, j)

    else

      $\lambda_{os} \Leftarrow$ Mutation(n_{total}, i)

    end if

    $\vec{x}_{\vec{\xi}_{os}} \Leftarrow \vec{-1}$

    if IsNotDuplicateChromosome$(gs, n_{total}, \lambda_{os}, \rho_\tau)$ **then**

      $n \Leftarrow$ Rearrange$(h, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \lambda_{os})$

      ReorderChromosome(h, λ_{os})

      if IsNotDuplicateChromosome$(gs, n_{total}, \lambda_{os}, \rho_\tau)$ **then**

        Calculate $C_{\lambda_{os}}, \vec{C}_{\lambda_{os}}$, UpdateFitnessOfPopulation$(f_m, f_f, \phi_{\lambda_{min}}, \phi_{\lambda_{max}}, \rho_\tau)$

        SortPopulation(ρ_τ)

        if $n = n_{total}$ **then**

          return

        end if

      end if

    end if

  end while

end for

Figure 4.8. Pseudocode for GA method.

structures

- Storing the best and the global best "particle orderings" for finding the best placement in the iterations
- Mutation of "particles", as proposed by Yarpiz [109]
- "Velocity" mirroring [109]
- Position clamping [109]
- Removing the items with the same size as the item that do not fit into the container in the PSO iterations, from the list to continue searching

The representations of the structures and our approach are presented below.

A "particle" i, ρ_i^p, is represented by $(\iota, w, l, \vec{B}_L, \vec{B}_U, \vec{x}^p, \vec{v}, O_\rho, \Psi_\rho)$, which are the item number of the "particle" in the original packing order, the width of mug, the length of mug, the lower bound vector for the minimum position, the upper bound vector for the maximum position, the position vector representing (x, y) coordinate of the mug center, the "velocity" vector for (v_x^p, v_y^p), the objective function value for the "particle", and the fitness parameters of the "particle", respectively. The lower bound for the "particle" is the lowest possible mug center, and it is found through adding the mug dimensions relative to the mug center to the lower limit for the tray, x_{Ltr}, y_{Ltr}. Similarly, for the upper bound, the relative sizes are subtracted from the upper tray limit, x_{Utr}, y_{Utr}.

In other words, the bounds for the "particle" i are:

$$\begin{aligned} \vec{B}_{L_{\rho_i}} &= (x_{Ltr} + l_{\rho_i}/2, y_{Ltr} + l_{\rho_i}/2) \\ \vec{B}_{U_{\rho_i}} &= (x_{Utr} - w_{\rho_i} + l_{\rho_i}/2, y_{Utr} - l_{\rho_i}/2) \end{aligned} \tag{4.17}$$

since the mug handle is facing the upper part of the tray in the X axis.

A "particle ordering" is used for representing the placement solution. For iteration τ, the "particle" ordering χ_τ^p (or named χ for simplicity) is represented as $(\vec{\rho}, O_\chi, \Psi_\chi)$, which denote the vector of all "particles", objective function value, fitness function parameters of the overall placement, respectively. At each iteration τ, the best "particle" ordering that has

the lowest cost, χ_b is found.

The update rules for the "velocity" and the position of a "particle" are:

$$\vec{v}^p_{\rho_i}(\tau+1) = \omega\vec{v}^p_{\rho_i}(\tau) + c_1 r_1 \times (\vec{x}^p_{\rho_{ib}}(\tau) - \vec{x}^p_{\rho_i}(\tau)) + c_2 r_2 \times (\vec{x}^p_{\rho_{gb}}(\tau) - \vec{x}^p_{\rho_i}(\tau)) \tag{4.18}$$

$$\vec{x}^p_{\rho_i}(\tau+1) = \vec{x}^p_{\rho_i}(\tau) + \vec{v}^p_{\rho_i}(\tau+1) \tag{4.19}$$

where c_1, c_2 are the acceleration coefficients, and r_1, r_2 are random numbers distributed in the interval [0, 1], and ω is the inertia weight. $\vec{x}^p_{\rho_{ib}}, \vec{x}^p_{\rho_{gb}}$ are the "particle"'s best position and position of the global best "particle", respectively.

The inertia weight (ω) is calculated as follows:

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{\tau}{\tau_{max}} \tag{4.20}$$

This approach may result in overlaps between items and also the items can leave the tray space, therefore, the following functions are defined, as described in [51, 52]:

$$\zeta_r(i) = \begin{cases} 0 & \text{if the item } i \text{ is inside the boundaries of the tray} \\ 1 & \text{otherwise} \end{cases} \tag{4.21}$$

where $\zeta_r(i)$ is the function for determining is an item is packed inside the tray.

$$\zeta_o(i,j) = \begin{cases} 0 & \text{if the items } i \text{ and } j \text{ are not overlapped} \\ 1 & \text{otherwise} \end{cases} \tag{4.22}$$

where $\zeta_o(i,j)$ checks if there are any overlaps in the placement, that is, if the placement is

valid. The objective function is found through these functions:

$$O(\vec{x}_i^p) = \frac{1}{1 + \sum_{i=1}^{k} \left\{ \zeta_r(i) + \sum_{j=1}^{k, i \neq j} \zeta_o(i,j) \right\}} \tag{4.23}$$

In order to compare the costs of two "particles", we define the placement parameters for a "particle" ϕ_ρ, which are the placed value of the "particle" in the tray n_ρ, the covered area (A_ρ), placement time (t_ρ), energy (E_ρ) and the sortedness (s_ρ). n_ρ is 1 if the "particle" is placed, 0 otherwise. In these parameters, the area of the current "particle", and the distance for moving the current "particle" from the counter-top to the tray are considered for finding the cost parameters and the associated cost. The overall placement parameters for the "particle ordering", ϕ_χ, namely the number of placed "particles" (n), the covered area (A), the placement time (t), the energy (E) and the sortedness (s) of the placement, are calculated as stated in Section 4.2.1.

Initially, each "particle" i is assigned a random position in the $[\vec{B}_{L_{\rho_i}}, \vec{B}_{U_{\rho_i}}]$ range. The velocities are randomly assigned within $[\vec{v}^p_{min}, \vec{v}^p_{max}]$ range. For each "particle", the objective function value, O_i, is set to 0, the cost (C_{ρ_i}) is set to 1, each of the cost parameters, $\vec{C}_{\rho_i}$, are set to 1. The "particle" best position and the global best position are set to (-1,-1), which represent that they are not on the tray. The "particle" and the global best cost and cost parameters are also set to 1, along with the objective function value, which is set to 0. The minimum placement parameters for the "particles", $\phi_{\rho_{min}}$, and the maximum placement parameters for the "particles", $\phi_{\rho_{max}}$, are set to zero.

For the "particle ordering", the objective function value, O_χ, is set to 0, the cost (C_χ) is set to 1, each of the cost parameters, $\vec{C}_\chi$, are set to 1. The minimum placement parameters for the overall placement, $\phi_{\chi_{min}}$, and the maximum placement parameters, $\phi_{\chi_{max}}$, are set to zero. These parameters are also initialized the same way for the best "particle ordering".

The neighboring solutions are found through applying the update rule. We also apply mutation in each iteration on each of the "particles" by swapping the coordinates of a "par-

ticle", that is, change the x to the y value and vice versa, as proposed by Yarpiz [109]. We use mirroring of "velocity", such that, if the "particle" reaches the boundaries of the container, the current "velocity" of the "particle" is multiplied by -1, such that the position of the "particle" will start decreasing, as proposed by Yarpiz [109]. Furthermore, the position vector of the "particle" is clamped, that is, the boundary value is assigned to the coordinate that exceeds the boundary value [109]. This approach allows keeping the "particle" within the boundaries of the container.

The procedure described above is shown in Figure 4.11, where $f_m, f_f, \rho'_{gb}, \rho_{lb}, \hat{\rho}_i$ denote fitness function method, fitness function type, previous global best "particle", local best "particle" and mutated "particle" i, respectively. lbg is 1 if the local best "particle" is greater than the global best "particle", otherwise it is 0.

We also implemented the improved PSO proposed by Shin et al [51, 52]. The update rule for the improved PSO is as presented there:

$$\vec{v}^{\,p}_{\rho_i}(\tau+1) = \omega\vec{v}^{\,p}_{\rho_i}(\tau) + c_1 r_1 \times (\vec{x}^{\,p}_{\rho_{ib}}(\tau) - \vec{x}^{\,p}_{\rho_i}(\tau)) + c_2 r_2 \times (\vec{x}^{\,p}_{\rho_{gb}}(\tau) - \vec{x}^{\,p}_{\rho_i}(\tau)) + c_3 r_3 \times (\vec{x}^{\,p}_{g2b}(\tau) - \vec{x}^{\,p}_{\rho_i}(\tau)) \tag{4.24}$$

where c_3 is an acceleration coefficient, and r_3 is a random number from in the interval [0, 1].

The procedure for the improved PSOp is given in Figure 4.12, where the "velocity" of a "particle" is updated with Eq. 4.24 according to a determined probability, PS. The inner PSOp procedure is presented in Figure 4.10. Finally, we modified the version of the PSOp used for the bin-packing problem, as described by Shin et al [51, 52]. We remove the items that have the same size as the item that creates an $O_\chi < 1$, that is, a solution with overlapping or out-of-bounds items, such that the iteration can continue with smaller items if there are any. The removed items are the items in the list that comes after the item k, which results in an undesired solution, including the item k. The overall procedure for PSOp is shown in Figure 4.9.

Input $h, \kappa, f_m, f_f, pso_m, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$

Output χ_{gb}

SortList(κ, Λ)

$\chi \Leftarrow \Lambda$

for $i \Leftarrow 1$ **to** n_{total} **do**

Calculate $\vec{B}_{L_i}, \vec{B}_{U_i}$

$\vec{x}_i^p \Leftarrow \vec{-1}$

$\iota_i \Leftarrow i$

end for

for $k \Leftarrow 1$ **to** n_{total} **do**

$C_\chi \Leftarrow 1, \vec{C}_\chi \Leftarrow \vec{1}, O_\chi \Leftarrow 0$

for $i \Leftarrow 1$ **to** k **do**

$C_{\rho_i} \Leftarrow 1, \vec{C}_{\rho_i} \Leftarrow \vec{1}, O_{\rho_i} \Leftarrow 0$

end for

for $i \Leftarrow 1$ **to** k **do**

$\vec{x}_i^p \Leftarrow$ random($[\vec{B}_{L_i}, \vec{B}_{U_i}]$)

end for

InnerPSOp(f_m, f_f, pso_m, k)

if $O_{\chi_b} < 1$ **then**

$n \Leftarrow k - 1$

$n_r \Leftarrow$ RemoveItemsWithSameSizeAsNotFittingItem

$n_{total} \Leftarrow n_{total} - n_r$

else

$\chi_{gb} \Leftarrow \chi_b$

$n \Leftarrow n + 1$

end if

end for

Figure 4.9. Pseudocode for PSOp.

4.2.5.2. PSOk. We modified the PSO method proposed by Domingo et al [53]. Our modifications to their approach are as follows:

- "Particle" (ρ_i^k) representation and the "particle ordering" structure (χ_τ^k)
- Initialization of keys according to the sort method
- Packing method
- Integration of the fitness functions and the fitness methods for the comparison of two structures
- Storing the best and the global best "particle orderings" for finding the best placement in the iterations
- "Velocity" mirroring [109]
- Position clamping [109]
- Addition of improved PSO

The structure of PSOk is very similar to PSOp. The costs and the placement parameters for each "particle" and the "particle ordering" are made in the same way as PSOp. The differences between the two approaches are the use of keys for determining the position of the "particle" and the use of the packing methods (BLF or LFB) for placing the items in PSOk, whereas, PSOp updates the actual positions of the "particles" through the "velocity" and also uses mutations to find the near-optimal solution faster. We have not utilized mutations for the PSOk, as mutation is costly in terms of time complexity, and the "velocity" changes the ordering of the items in the packing, which is similar to the mutation. PSOp is a slower algorithm as its structure requires it to do more iterations to find non-overlapping solutions.

A "particle" i, ρ_i^k, is represented by $(\iota_i, w_i, l_i, \vec{x}_i^k, \Theta_i^k, v_i^k, \Psi_\tau)$, which are the item number of the "particle" in the original packing order, the width of mug, the length of mug, the key of the "particle", the "velocity" of the key, and the fitness parameters of the "particle".

The update rules for the "velocity" ($v_{\rho_i}^k$) and the key ($\Theta_{\rho_i}^k$) of a "particle" i are:

$$v_{\rho_i}^k(\tau+1) = \omega v_{\rho_i}^k(\tau) + c_1 r_1 \times (\Theta_{\rho_{ib}}^k(\tau) - \Theta_{\rho_i}^k(\tau)) + c_2 r_2 \times (\Theta_{\rho_{gb}}^k(\tau) - \Theta_{\rho_i}^k(\tau)) \qquad (4.25)$$

Input f_m, f_f, pso_m, k

Output χ_b

$\phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}} \Leftarrow 0$

for $\tau \Leftarrow 1$ **to** τ_{max} **do**

Calculate ω

Calculate $C_\chi, \vec{C}_\chi, O_\chi$

if $\tau = 1$ **then**

$\phi_{\chi_{min}} \Leftarrow \phi_{\chi_{max}}$

else

UpdateFitness($f_m, f_f, \phi_{\chi_{min}}, \phi_{\chi_{max}}, \chi_b$)

end if

for $i \Leftarrow 1$ **to** k **do**

$O_{\rho_i} \Leftarrow O_\chi$

if ρ_{ib} is placed within the tray **then**

UpdateParticleFitness($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \rho_{ib}$)

else

$\phi_{\rho_{ib}} \Leftarrow \phi_{\rho_{max}}$, $C_{\rho_{lb}} \Leftarrow 1$, $\vec{C}_{\rho_{lb}} \Leftarrow \vec{1}$

end if

end for

UpdateParticleFitness($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$)

if $k \geq 2$ **and** $pso_m = 1$ **then**

ImprovedPSOp($\omega, f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$)

else

NormalPSOp($f_m, f_f, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}$)

UpdateVelocityPSOp(ω, ρ_{gb}, χ), UpdatePositionPSOp(χ)

end if

Calculate t_c

end for

$\chi \Leftarrow \chi_b$

Figure 4.10. Pseudocode for InnerPSOp.

Input $f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$

Output $\rho_{lb}, lbg, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$

$C_{\rho_{lb}} \Leftarrow 1, \vec{C}_{\rho_{lb}} \Leftarrow \vec{1}, O_{\rho_{lb}} \Leftarrow 0$

$lbg \Leftarrow 0$

for $i \Leftarrow 1$ **to** k **do**

  Mutate ρ_i to $\hat{\rho}_i$

  Calculate $C_{\hat{\rho}_i}, \vec{C}_{\hat{\rho}_i}, O_{\hat{\rho}_i}$

  UpdateParameters($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}$)

  if $O_{\hat{\rho}_i} > O_{\rho_i}$ **or** $O_{\hat{\rho}_i} = O_{\rho_i}$ **and** IsCostDecreased($f_m, f_f, \hat{\rho}_i, \rho_i$) **then**

    $\rho_i \Leftarrow \hat{\rho}_i$, update χ, $O_\chi \Leftarrow O_{\hat{\rho}_i}$

    UpdateParticleFitness($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$)

  end if

  if $O_{\rho_i} > O_{\rho_{ib}}$ **or** $O_{\rho_i} = O_{\rho_{ib}}$ **and** IsCostDecreased($f_m, f_f, \rho_i, \rho_{ib}$) **then**

    $\rho_{ib} \Leftarrow \rho_i$

  end if

end for

Calculate $C_\chi, \vec{C}_\chi$

UpdateFitness($f_m, f_f, \phi_{\chi_{min}}, \phi_{\chi_{max}}, \chi_b$)

for $i \Leftarrow 1$ **to** k **do**

  if $O_{\rho_{ib}} > O_{\rho_{lb}}$ **or** $O_{\rho_{ib}} = O_{\rho_{lb}}$ **and** IsCostDecreased($f_m, f_f, \rho_{ib}, \rho_{lb}$) **then**

    $\rho_{lb} \Leftarrow \rho_{ib}$

  end if

end for

if $O_{\rho_{lb}} > O_{\rho_{gb}}$ **or** $O_{\rho_{lb}} = O_{\rho_{gb}}$ **and** IsCostDecreased($f_m, f_f, \rho_{lb}, \rho_{gb}$) **then**

  $\rho'_{gb} \Leftarrow \rho_{gb}$, $\rho_{gb} \Leftarrow \rho_{lb}$, $lbg \Leftarrow 1$

end if

if $O_\chi > O_{\chi_b}$ **or** $O_\chi = O_{\chi_b}$ **and** IsCostDecreased(f_m, f_f, χ, χ_b) **then**

  $\chi_b \Leftarrow \chi$

end if

Figure 4.11. Pseudocode for NormalPSOp.

Input $\omega, f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$
Output $\chi, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$
NormalPSOp($f_m, f_f, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}$)
if $lbg = 1$ **then**
  $\rho_{g2b} \Leftarrow \rho'_{gb}$
  for $i \Leftarrow 1$ **to** k **do**
    if $\rho_i \neq \rho_{lb}$ **then**
      if $O_{\rho_{ib}} > O_{\rho_{g2b}}$ **or** $O_{\rho_{ib}} = O_{\rho_{g2b}}$ **and** IsCostDecreased($f_m, f_f, \rho_{ib}, \rho_{g2b}$) **then**
        $\rho_{g2b} \Leftarrow \rho_{ib}$
      end if
    end if
  end for
else
  if $O_{\rho_{lb}} > O_{\rho_{g2b}}$ **or** $O_{\rho_{lb}} = O_{\rho_{g2b}}$ **and** IsCostDecreased($f_m, f_f, \rho_{lb}, \rho_{g2b}$) **then**
    $\rho_{g2b} \Leftarrow \rho_{lb}$
  end if
end if
Pick random $r \in [0, 1]$
if $r > PS$ **then**
  UpdateVelocityPSOp(ω, ρ_{gb}, χ)
else
  UpdateVelocityImprovedPSOp($\omega, \rho_{gb}, \rho_{g2b}, \chi$)
end if
UpdatePositionPSOp(χ)

Figure 4.12. Pseudocode for ImprovedPSOp.

$$\vec{x}^k_{\rho_i}(\tau+1) = \Theta^k_{\rho_i}(\tau) + \vec{v}^k_{\rho_i}(\tau+1) \tag{4.26}$$

The update rule for the "velocity" of a "particle" for the improved PSO is presented below:

$$\begin{aligned} v^k_{\rho_i}(\tau+1) = \omega v^k_{\rho_i}(\tau) + c_1 r_1 \times (\Theta^k_{\rho_{ib}}(\tau) - \Theta^k_{\rho_i}(\tau)) + c_2 r_2 \times (\Theta^k_{\rho_{gb}}(\tau) - \Theta^k_{\rho_i}(\tau)) + \\ c_3 r_3 \times (\Theta^k_{\rho_{g2b}}(\tau) - \Theta^k_{\rho_i}(\tau)) \end{aligned} \tag{4.27}$$

Similar to PSOp, a "particle ordering" is used for representing the placement solution. For iteration τ, the "particle ordering" χ^k_τ is represented as $(\vec{\rho}, \Psi_\chi)$, which denote the vector of all "particles", and the fitness parameters of the overall placement, respectively. The best "particle ordering" with the lowest cost, χ_b, is found at each iteration.

Initially, each "particle" i is assigned a random key in the [0,1] range. The keys are sorted according to the sort method (κ) to assign them to the "particles". The velocities are randomly assigned within $[v^k_{min}, v^k_{max}]$ range.

The packing of items is conducted as described in Section 4.2.2, which determines which of the items are packed in the tray and the coordinates of the packed items inside the tray.

χ^k_τ, ρ^k are denoted as χ and ρ in the pseudocodes presented for PSOk, which are presented in Figure 4.13, Figure 4.14, Figure 4.15 and Figure 4.16.

Input $h, \kappa, f_m, f_f, pso_m, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$

Output χ_{gb}

SortList(κ, Λ)

$\chi \Leftarrow \Lambda$

for $i \Leftarrow 1$ **to** n_{total} **do**

  $\vec{x}_i^p \Leftarrow \vec{-1}$

  $\iota_i \Leftarrow i$

end for

$C_\chi \Leftarrow 1, \vec{C}_\chi \Leftarrow \vec{1}$

for $i \Leftarrow 1$ **to** n_{total} **do**

  $C_{\rho_i} \Leftarrow 1, \vec{C}_{\rho_i} \Leftarrow \vec{1}$

end for

InitializeKeys(κ, n_{total})

if κ is Random-Ordering **then**

  SortItemsAccordingToKeys(n_{total}, Λ)

end if

InnerPSOk($h, \kappa, f_m, f_f, pso_m, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$)

$n \Leftarrow n_{\chi_{gb}}$

Figure 4.13. Pseudocode for PSOk.

4.3. Manipulation Planning

In order to find the configuration space of the arm, for each determined z position upper and lower x limits were found, and the maximum and minimum y limits were found accordingly. The limits were found using *Gazebo*: the maximum and minimum physical limits were given as the position for the end effector in both side grasp and top grasps, and the kinematics were calculated through *ROS MoveIt!* and *KDL*. If a valid kinematics solution can be calculated, the upper limit was increased by 1 cm and decreased otherwise. For side grasps, the end effector could reach a range of 22 cm in the forward direction (in x), and 82 cm in the horizontal direction (in y), which correlates with the physical arm limits for the given grasp position. For the top grasps, the end effector could reach a range of 22 cm in x, and a maximum of 82 cm in y in the lower x limit, however, it can only reach up to 40 cm in y in the upper x limit, which brought about two choices: either using a smaller size tray or only using side grasps for placing objects on the tray, since the arm will not be able to reach the upper corner of the tray, if the tray size was found using side grasps. In order to find the maximum number of mugs that can be placed in a tray, we used the tray with the maximum size. Due to the fact that the arm does not have a wrist yaw joint, the gripper needs to be at an angle α given by Eq. 4.28 in order to reach (x,y) position for the side grasps.

$$\alpha = \arctan(y_g/x_g) \tag{4.28}$$

Hence, from geometry, we can derive the equations of the gripper x, y positions as follows:

$$r_m = \frac{l_m}{2} \tag{4.29}$$

$$\Delta_{mg} = \sqrt{(r_m - w_g)^2 + l_g^2} \tag{4.30}$$

$$\Delta_{ma} = \sqrt{(x_m - x_a)^2 + (y_m - y_a)^2} \tag{4.31}$$

Input $h, \kappa, f_m, f_f, pso_m, k, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda, \chi$

Output χ_{gb}

$\phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}} \Leftarrow 0$

for $\tau \Leftarrow 1$ **to** τ_{max} **do**

Calculate ω

Rearrange($h, n_{total}, x_{Ltr}, y_{Ltr}, x_{Utr}, y_{Utr}, \Lambda$)

Calculate $C_\chi, \vec{C}_\chi$

if $\tau = 1$ **then**

$\phi_{\chi_{min}} \Leftarrow \phi_{\chi_{max}}$

else

UpdateFitness($f_m, f_f, \phi_{\chi_{min}}, \phi_{\chi_{max}}, \chi_b$)

end if

for $i \Leftarrow 1$ **to** k **do**

if ρ_{ib} is placed within the tray **then**

UpdateParticleFitness($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \rho_{ib}$)

else

$\phi_{\rho_{ib}} \Leftarrow \phi_{\rho_{max}}$, $C_{\rho_{lb}} \Leftarrow 1$, $\vec{C}_{\rho_{lb}} \Leftarrow \vec{1}$

end if

end for

UpdateParticleFitness($f_m, f_f, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$)

if $k \geq 2$ **and** $pso_m = 1$ **then**

ImprovedPSOk($\omega, f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}, \phi_{\chi_{min}}, \phi_{\chi_{max}}$)

else

NormalPSOk($f_m, f_f, \rho_{gb}, \rho'_{gb}, \rho_{g2b}, \phi_{\rho_{min}}, \phi_{\rho_{max}}$)

UpdateVelocityPSOk(ω, ρ_{gb}, χ), UpdateKeyPSOk(χ)

end if

SortItemsAccordingToKeys(n_{total}, Λ), Calculate t_c

end for

$\chi_{gb} \Leftarrow \chi_b$

Figure 4.14. Pseudocode for InnerPSOk.

Input $f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$

Output $\rho_{lb}, lbg, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$

$C_{\rho_{lb}} \Leftarrow 1, \vec{C}_{\rho_{lb}} \Leftarrow \vec{1}$

$lbg \Leftarrow 0$

for $i \Leftarrow 1$ **to** k **do**

  if IsCostDecreased($f_m, f_f, \rho_i, \rho_{ib}$) **then**

    $\rho_{ib} \Leftarrow \rho_i$

  end if

end for

for $i \Leftarrow 1$ **to** k **do**

  if IsCostDecreased($f_m, f_f, \rho_{ib}, \rho_{lb}$) **then**

    $\rho_{lb} \Leftarrow \rho_{ib}$

  end if

end for

if IsCostDecreased($f_m, f_f, \rho_{lb}, \rho_{gb}$) **then**

  $\rho'_{gb} \Leftarrow \rho_{gb}$, $\rho_{gb} \Leftarrow \rho_{lb}$, $lbg \Leftarrow 1$

end if

if IsCostDecreased(f_m, f_f, χ, χ_b) **then**

  $\chi_b \Leftarrow \chi$

end if

Figure 4.15. Pseudocode for NormalPSOk.

Input $\omega, f_m, f_f, \chi, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$
Output χ
NormalPSOk($f_m, f_f, \rho_{gb}, \rho'_{gb}, \rho_{g2b}$)
if $lbg = 1$ **then**
  $\rho_{g2b} \Leftarrow \rho'_{gb}$
  for $i \Leftarrow 1$ **to** k **do**
    if $\rho_i \neq \rho_{lb}$ **then**
      if IsCostDecreased($f_m, f_f, \rho_{ib}, \rho_{g2b}$) **then**
        $\rho_{g2b} \Leftarrow \rho_{ib}$
      end if
    end if
  end for
else
  if IsCostDecreased($f_m, f_f, \rho_{lb}, \rho_{g2b}$) **then**
    $\rho_{g2b} \Leftarrow \rho_{lb}$
  end if
end if
Pick random $r \in [0, 1]$
if $r > PS$ **then**
  UpdateVelocityPSOk(ω, ρ_{gb}, χ)
else
  UpdateVelocityImprovedPSOk($\omega, \rho_{gb}, \rho_{g2b}, \chi$)
end if
UpdateKeyPSOk(χ)

Figure 4.16. Pseudocode for ImprovedPSOk.

$$\theta = \arctan\left(\frac{r_m - w_g}{l_g}\right) \tag{4.32}$$

$$\beta' = \arctan\left(\frac{|y_m - y_a|}{x_m - x_a}\right) \tag{4.33}$$

$$\gamma' = \arcsin\left(\frac{r_m - w_g}{\Delta_{ma}}\right) \tag{4.34}$$

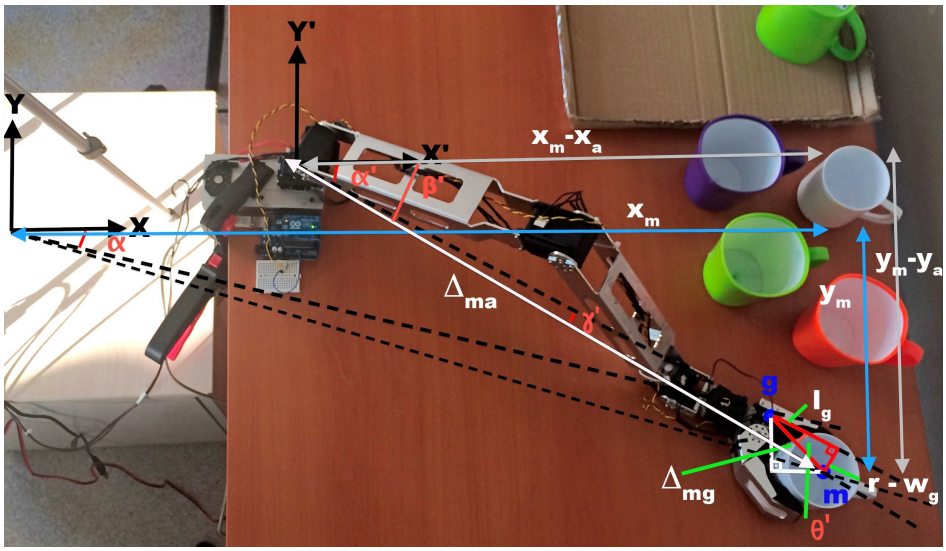
$$\alpha' = \beta' - \gamma' \tag{4.35}$$

$$x_g = x_m - \Delta_{mg}\cos(\alpha' + \theta) \tag{4.36}$$

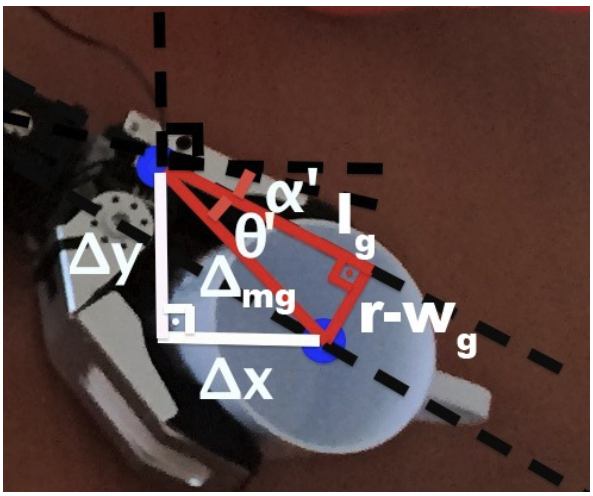
$$y_g = y_m \mp \Delta_{mg}\sin(\alpha' + \theta) \tag{4.37}$$

where x_m and y_m are the position of the center of the mug brim, l_m is the length of the gripper part that is movable, i.e. "fingers", w_m is the width of the gripper "thumb", and r_m is the radius of the mug brim. $\alpha', \gamma', \beta', \theta'$ are with respect to the arm base coordinate frame, the angle of the target position of the gripper, the angle between the target position of the gripper and the mug center, the angle of the mug center, and the angle between gripper and mug center and the length of the gripper, respectively, whereas α is with respect to the world frame. θ is the angle The x, y positions of the mug and the gripper are given with respect to the world frame, e.g. x_a and y_a are the x, y positions of the arm base with respect to the world frame. Δ_{mg} and Δ_{ma} represent the distance of the mug center to the gripper and arm base, respectively. The gripper equations hold both for the side and the upside-down side grasps. In Eq. 4.37 the difference is added for the side grasp, and subtracted for the upside-down grasp, because the position of the gripper "thumb" changes from left to right. The world frame is defined at $(x, y, z) = (0, 0, 0)$ as described in Section 3.2.3.

Figure 4.17 shows the geometry of the equations.



(a)



(b)

Figure 4.17. Frame diagrams for the side grasp.

In order to find the equations for the tray limits, we need to find the equations for the relation between the mug center and the gripper target position.

$$x_m = x_g + \Delta_{mg}\cos(|\alpha| + \theta) \tag{4.38}$$

$$y_m = y_g \pm \Delta_{mg} \cos(|\alpha| + \theta) \tag{4.39}$$

In Eq. 4.39 the difference is added if the gripper is upright, and subtracted if the gripper is upside-down. By using these mug equations for the upside-down gripper, we can derive the equations to find the tray limits.

$$x_{Ltr} = x_{Lg} + \Delta_{mg} \cos(|\alpha_{Lx}| + \theta) - min(w_r) \tag{4.40}$$

$$x_{Utr} = x_{Ug} + \Delta_{mg} \cos(|\alpha_{Ux}| + \theta) + min(w_m) - min(w_r) \tag{4.41}$$

$$y_{Utr} = y_{Ug} + \Delta_{mg} \sin(|\alpha_{Ux}| + \theta) + min(w_r) \tag{4.42}$$

$$y_{Ltr} = 0.132 \tag{4.43}$$

$$\alpha_{Lx} = \arctan(y_{Ug}/x_{Lg}) \tag{4.44}$$

$$\alpha_{Ux} = \arctan(y_{Ug}/x_{Ug}) \tag{4.45}$$

where $x_{Ltr}, x_{Utr}, y_{Ltr}, y_{Utr}, x_{Lg}, x_{Ug}, y_{Lg}, y_{Ug}$ are the lower and upper x and y limits for the tray and the gripper. The minimum mug width and length are used to find the tray limits. If the limits were to be found using the maximum width or length, when a smaller mug is to be placed, a mug position with a lower x_g or larger y_g than the gripper limits will be calculated, which would result in an infeasible position for the gripper. As x_g decreases, α increases. Hence, if α_{Lx} was used to find y_{Utr}, an infeasible y_g would be calculated at the x_{Ug} limit. Therefore, α_{Ux} is used to find the y_{Utr}. y_{Ltr} was chosen to be 13.2 cm, in order to allow more mugs to be used in our dataset. The tray size was found using these equations to be 33.2 cm

in length (in x) and 37.4 cm in width (in y). As a result of using the tray size for side grasps, a rotational correction after the placement on the tray was not possible, since the arm does not have a wrist yaw joint and the arm cannot reach the upper corner of the tray for grasps from the top.

We did not use collision detection for the kinematics, since the computational time for planning a collision-free path by *ROS MoveIt!* was longer than 1 minute for each move, which made the process very time-consuming. Therefore, in order to avoid collision with other mugs, the mugs are moved by the gripper in $x-y$ space at a z position, where there is at least 3 cm difference between the top of the mug on the counter-top and the bottom of the mug that the gripper is holding. In order to turn a mug upside down on the tray, the gripper needs to grasp from the side of the mug. However, if the placement algorithm is not "closest-item-first", then it would not be possible to grasp a mug which is behind other mugs. Therefore, either a mug needs to be grasped from the top and placed on an empty area in front of the robot, so that each mug can be placed in the "Top-Left-Fill" (TLF) or "Left-Top-Fill" (LTF) order or the closest mugs to the robot should be placed first on the calculated positions on the tray, not maintaining the TLF or LTF placement order, which would reduce the need for grasping the mug from the top and placing at an empty area. The closest mug is defined as the mug which is at a x position closest to the x position of the robot, with a y position closest to the y position of the robot. The mugs on the closest x positions are to be placed first, in a bottom-left order, that is, starting from the bottom left-most mug, if the mugs are on the right of the robot or in a bottom-right order, that is, starting from the bottom right-most mug, if the mugs are on the left. However, placing the closest-item-first approach is infeasible by the robot, since a previously placed mug can block the path of the gripper, similar to the case in Figure 4.4.

For example, if there is a mug on the tray to the right diagonal of the mug to place, that is, at a lower x and lower y position, the gripper cannot descend to the necessary position to place the mug. Thus, in placing the closest-item-first approach the mugs should be placed on the tray by grasping from the top, which would require the mugs to be turned upside down on their counter-top positions. However, the configuration space for the arm in the top grasp is a smaller area, which would entail a smaller sized tray, which was undesirable. Hence, we

used the first approach to place the mugs on an empty area before placing them on the tray. The empty area can be a sink in a real life setting, and the mugs can be turned upside down on the sink, in order to empty any liquid within the mug. We also tried to turn the mugs upside down above the tray in order to empty any liquid into the tray as a possible solution, however, the arm was not able to reach high z positions at x, y positions farther from the arm base, hence the mug in manipulation collided with the other mugs on the tray while it was being moved to its final position on the tray. Therefore, we chose to turn the mugs upside-down on the empty area. The manipulation planning flow diagram is presented in Figure 4.18.

4.3.1. Recognizing Grasps

We determined thresholds for the FSR and the gripper aperture, in order to detect if a grasp is successful. The gripper closes until the value from the FSR is above the threshold. If the gripper aperture reaches a value less than a determined threshold value, the grasp is unsuccessful. Therefore, the object recognition algorithm is run, in order to find the lost mug. It checks if all of the mugs are at respective positions that they were at the beginning of the previous state. If a mug is within 1 cm margin of its previous positions, it is defined to be “known”. If there is only one “unknown” mug, it means that the mug that the gripper could not grasp is found. Therefore, its position is updated, and the manipulation planning state changes to the initial starting position and the mug is placed on the empty area again to be grasped from the side. If there are more than one “unknown” mugs, the mugs are sorted according to their closeness to the position of the remaining “unknown” mugs, and assigned accordingly. If a mug is “lost”, that is, the detected mugs on the counter-top are less than the previously detected mug set on the counter-top minus the placed mugs, then the placement algorithm is re-run, in order to change the placement inside the tray according to the “new” set. The placed mugs inside of the tray are preserved in the plan of the placement, in other words, the placement planning is made for the remaining space within the tray.

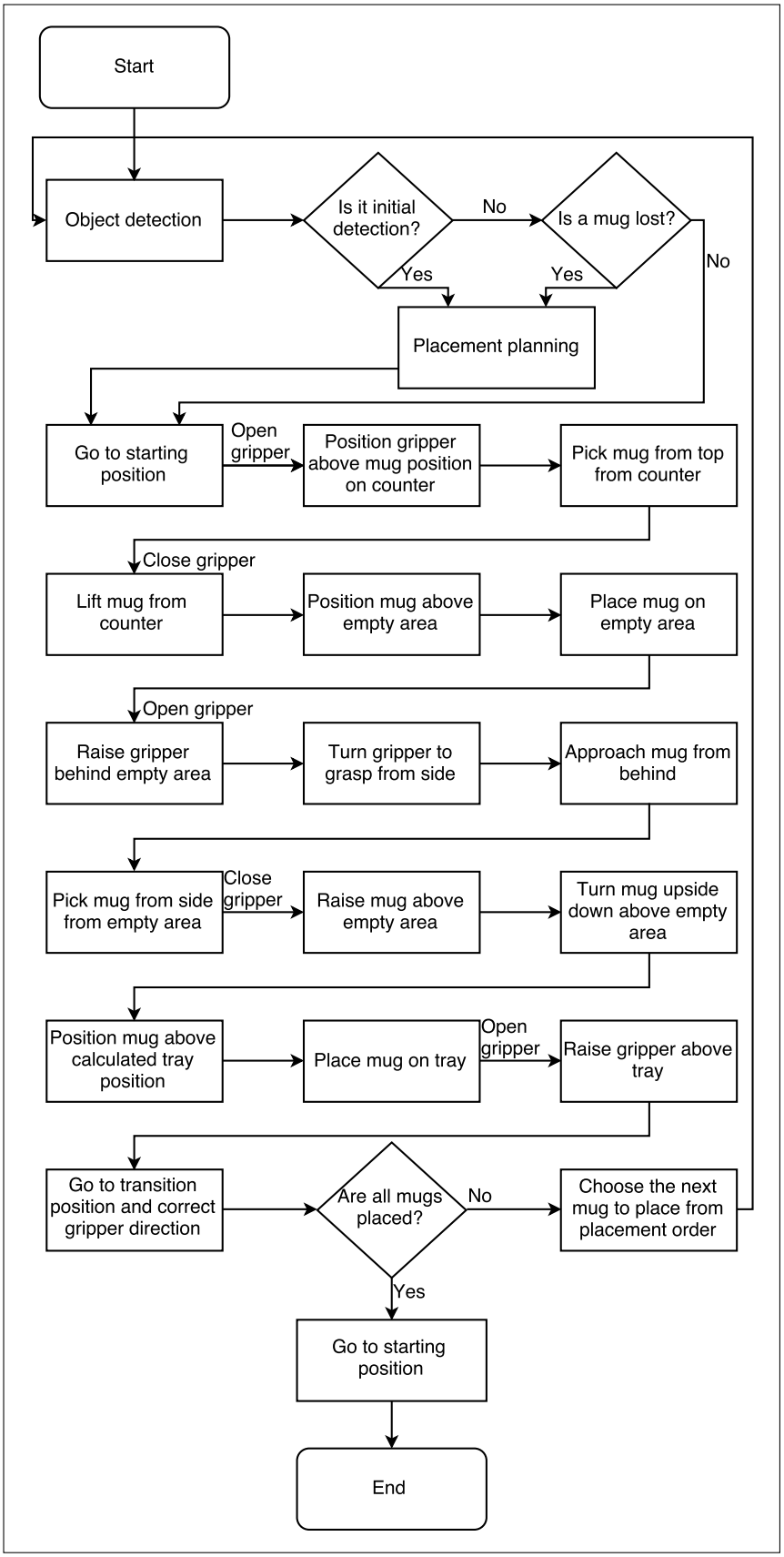


Figure 4.18. Manipulation planning flow diagram.

5. RESULTS AND DISCUSSION

We are using a 5 DOF robot arm, as stated in previous chapters, which is fixed to the table. *ASUS Xtion Pro Live* RGB-D is placed at 170 cm above the ground, 70 cm away from the table, and the camera pitch is 55 degrees. The robot experiments are conducted through running the code on the computer, which starts the servomotors, *ROS MoveIt!*, *openni_launch*, kinematics planner node, object recognition node and the placement planning node.

We use two types of mugs with different colors and shapes in our tests, which are lightweight so that the arm can lift them. Mug type one (big mug) is 9.1 cm in length, 12 cm in width, 10.6 cm in height, and weighs 200 grams. Mug type two (small mug) is 7.4 cm in length, 10 cm in width, 8 cm in height and 100 g in weight. The mug types are shown in Figure 5.1. We use a 37.4 cm width x 33.2 cm length rectangular wooden tray for our experiments.

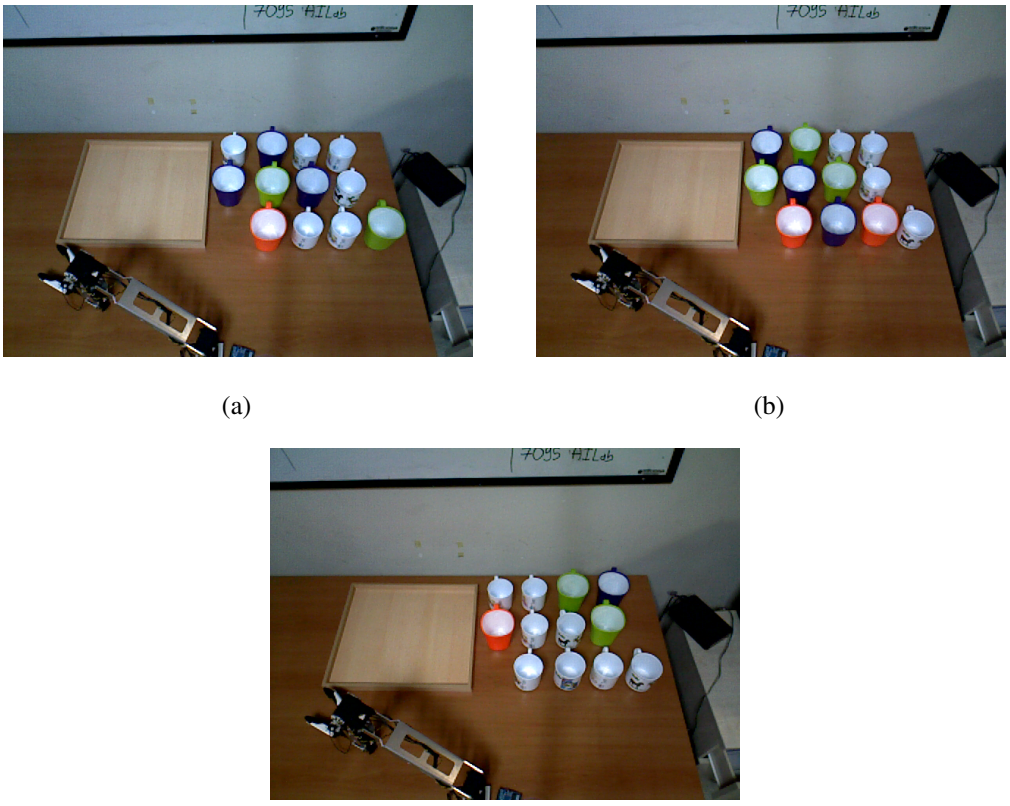
(a) (b) (c)

(d) (e) (f)

Figure 5.1. Mugs in the test set: (a)-(c) mug 1, (d)-(f) mug 2.

We have a total of 12 mugs in our robot experiment setting. There are three types of datasets for the robot experiments. The first set consists of six big and six small mugs. The

second dataset has eight big and four small mugs, and the final dataset has four big and eight small mugs. The RGB images of the settings as seen by the RGB-D camera are shown in Figure 5.2.



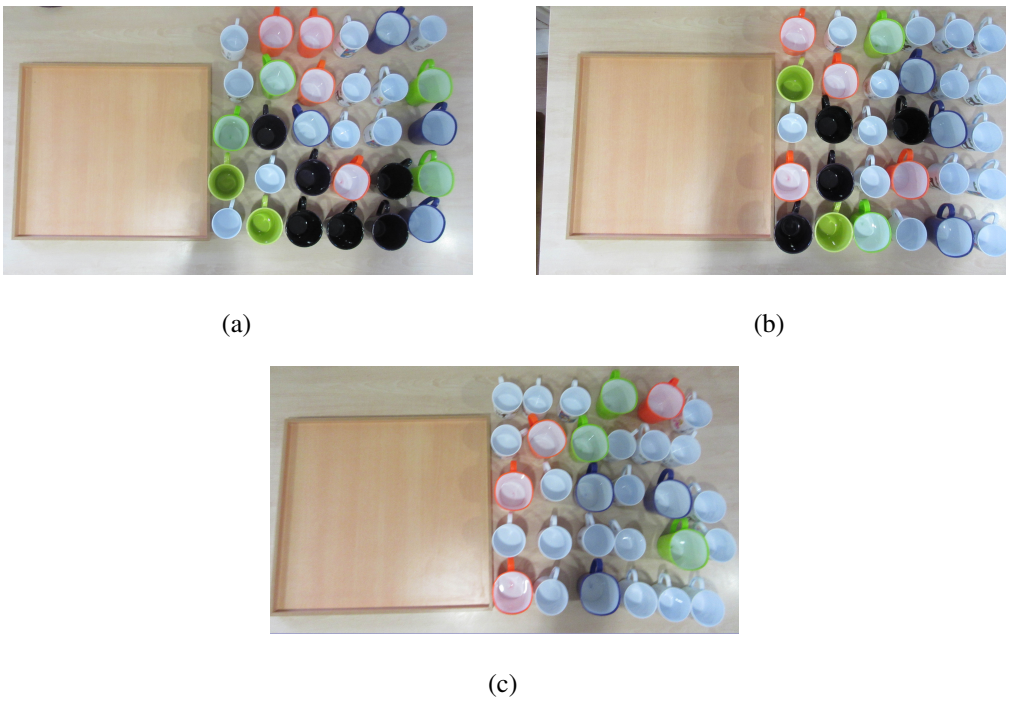
(a) (b)

(c)

Figure 5.2. The experimental datasets with 12 mugs: (a) six big, six small mug, (b) eight big, four small mugs, (c) four big, eight small mugs.

In order to examine more realistic results, we use three more types of datasets with 30 mugs for the experiments with people and the simulation experiments. The fourth set has 20 big and 10 small mugs, the fifth has 15 big and 15 small mugs, the final dataset has 10 big and 20 small mugs. We maintained the ratio of 1:1 and 2:1 as in the first three datasets. A bigger tray of 52 cm width x 47.5 cm length is used, which corresponds to a real dishwasher tray size. The datasets for the experiments with people are shown in Figure 5.3.

In this chapter, we present the results for object recognition in Section 5.1. Following



(a) (b)

(c)

Figure 5.3. The experimental datasets with 30 mugs: (a) 20 big, 10 small mug, (b) 15 big, 15 small mugs, (c) 10 big, 20 small mugs.

this section, the preliminary and user studies and their results are explained in Sections 5.2.1 and 5.2.2, respectively. The observational study and the annotation framework, which provided the weights and the ranks for our cost functions are given in Section 5.2.3 and 5.2.4, respectively. The resulting cost functions for the weighted-sum cost function are shown in Section 5.2.5, and the cost functions for the ranking method are presented in Section 5.2.6. Finally, we give the results for our experiments in Section 5.3. In this section, we present the tolerance test for finding the optimum tolerance amount between two mugs for our robot in Section 5.3.1. We compare our proposed methods to each other and to the sorted-size methods in the simulation in Section 5.3.2.

5.1. Object Recognition

The object recognition algorithm recognizes the small and big mugs separately. The confidence level above 50% is considered as a correct recognition. However, because the shapes of the two objects are similar, a small mug can be recognized as a big mug, and vice

versa. There are also other false positives in which a mug is detected at a location where there is no mug. The examples of the correct recognitions of the items and the false positives are shown in Figure 5.4.

5.2. Cost Function

In order to find the parameters and the weights of our cost function to use in the optimization methods, we conducted a preliminary study, a user study, an observational study and used an annotation system as described in Section 4.2.1. The results of these studies and the discussion on the results are presented in Sections 5.2.1 - 5.2.4.

5.2.1. Preliminary Study

We wanted to propose a system that can place the dishes according to the preferences of people, hence, we made a preliminary study in order to find the criteria that people consider, as described in Section 4.2.1.1. From these criteria, we can derive the parameters that we can optimize, and use them in a cost function for the comparison of two placement solutions in the packing algorithms. The preliminary study was conducted by asking our participants in person an open ended question to describe the criteria that they consider as important for placing dishes into the dishwasher. The study group for our preliminary study involved 31 women and 19 men: 17 people with the age range of 18-25, 9 people with the age range of 26-35, 12 people with the age range of 36-50, 6 people with the age range of 51-65 and 6 people older than 65. The results of the preliminary study are presented in Table 5.1.

In our project, the criteria (1), (3), (9), (11), (13) and (15) are satisfied. There is no stacking of mugs since the tray is modeled in 2D in the placement problem and all mugs are placed at the same Z coordinate on the flat surface of the tray, in our physical world experiments. The robotic arm releases the mugs slightly above the tray in order to place them, i.e. it does not drop them from above the tray, hence, the possibility of breaking the mugs is reduced. Also, we use a grasp planning which involves picking the mugs from the top in the clutter, in order to prevent the arm from pushing other mugs. We are only concerned with placing mugs in our project, hence, we are only using the upper tray. We are

Figure 5.4. Object recognition results and the examples of the false positives: (a) recognized big mug, (b) recognized small mug recognized, (c) small mug is recognized as a big mug, (d) big mug is recognized as a small mug, (e) a small mug recognized at a position where there is no mug.

Table 5.1. Results of the preliminary study.

Answer	Count
1. The dishes should not be stacked during placement, and the dishes should be positioned in the tray to be cleaned properly	26
2. Placing the dishes according to dish type, i.e. forks and knives should be in separate cells, the mugs should be separated from glasses	19
3. The dishes should not break during the placement	14
4. Number of placed dishes	13
5. Rinsing the dishes before placement	12
6. Placing the dishes according to dish size, i.e. bigger dishes should be separated from smaller ones	9
7. The fragile dishes should not be placed	8
8. Tall dishes that could collide with the spray arm should not be placed	8
9. The mugs and glasses should be placed on the upper tray, and plates, cups, and cutlery should be on the lower tray	7
10. The total placement time	5
11. The mugs and glasses should be upside-down in the tray and the sharp edge of the knives should be facing downwards	5
12. The electrical energy used by the robot	4
13. The mugs should have some spaces between them	4
14. The greasy dishes should be placed apart from the other dishes	3
15. The mugs should be turned upside-down on the tray	1

also using a small tolerance amount between the mugs within the tray, which was optimized through tolerance tests in our project. Furthermore, the mugs are turned upside down to be placed in the tray, and the turning operation is conducted on an empty area, which could be a sink in a real life setting, in order to prevent any liquid from spilling on the kitchen floor.

In order to conduct the user study, we determined the criteria that we could optimize using our optimization packing methods, therefore, we used (4), (6), (10) and (12) in our next questionnaire. In our study, we are concerned with placing only mugs on the dishwasher tray, hence, we omitted (2) and (8) from our user study. We cannot detect the fragility or the greasiness of a mug, hence, we did not include (7) and (14) in our cost function. Also, we do not have a water-resistant arm, hence (5) was invalid for our study.

5.2.2. User Study

For the user study described in Section 4.2.1.1, 258 people (121 women and 137 men) participated in our anonymous online questionnaire given in Figure B.1 in Appendix B. The majority of our participants were between 18-25 years old. One person younger than 18, 132 people with the age range of 18-25, 81 people with the age range of 26-35, 25 people with the age range of 36-50, 17 people with the age range of 51-65 and two people older than 65 participated in our study. Since it was an online survey, we could not control the number of people for the age range, so we had fewer answers from people who were older than a certain age. The results of the user study are presented in Table 5.2. The parameters (1), (2), (3) and (4) are represented by the number of placed mugs (n), electrical energy (E), placement time (t) and sortedness (s) parameters in the cost function. According to the results from the user study, the weights of the parameters of the user preference-based cost function (C_u), as defined in Eq. 4.9 for the weighted-sum method, are calculated using Eq. 4.7 in Section 4.2.1.5, and these weights are presented in Table 5.3. Along with the overall weights of the parameters, we analyzed the effect of gender, age, working status, student status, existence of a dishwasher in the household and the frequency of dishwasher usage.

These weights show that the most important criterion for the placement of the dishes in the overall and also for all types of participants is the number of mugs/dishes placed.

Table 5.2. The overall results of the user study of 258 participants.

Parameter	**Rank 1**	**Rank 2**	**Rank 3**	**Rank 4**	**Not Important**
1. Using the maximum capacity of the dishwasher	133	66	36	17	6
2. Spending small amount of electrical energy during placement	43	96	66	32	21
3. Placing the dishes in a short time	52	58	101	27	20
4. Placing the dishes according to dish size	26	37	36	82	77

Table 5.3. The weights of the user-preference based cost function (C_u) using the results of the user study.

Categories	**Attributes**	W_n	W_t	W_E	W_s	Participants
Overall Weight		0.338	0.252	0.258	0.152	258 (100%)
Gender	Women	0.335	0.254	0.262	0.149	121 (46.9%)
	Men	0.341	0.251	0.253	0.155	137 (53.1%)
Age Range	Younger than 18	0.3	0.2	0.4	0.1	1 (0.4%))
	18-25	0.338	0.246	0.245	0.171	132 (51%))
	26-35	0.348	0.263	0.262	0.126	81 (31%)
	36-50	0.352	0.217	0.291	0.139	25 (10%)
	51-65	0.297	0.297	0.290	0.116	17 (7%)
	Older than 65	0.361	0.255	0.236	0.148	2 (1%)
Working Status	Working	0.346	0.261	0.248	0.145	164 (64%)
	Not-working	0.329	0.237	0.27	0.164	94 (36%)
Student Status	Student	0.361	0.255	0.236	0.148	162 (63%)
	Not a student	0.303	0.258	0.28	0.159	96 (37%)
Dishwasher Ownership	Owns a dishwasher	0.339	0.249	0.266	0.146	230 (89%)
	Does not have a dishwasher	0.345	0.277	0.176	0.202	28 (11%)
Usage Frequency	Once every two weeks or less	0.326	0.248	0.233	0.193	54 (21%)
	Once a week	0.357	0.244	0.258	0.141	92 (36%)
	Two-three times a week	0.326	0.251	0.27	0.153	78 (30%)
	More than three times a week	0.343	0.261	0.254	0.142	34 (13%)

This suggests that the optimization algorithms would provide more preferable results than other heuristics. The least significant criterion is placing dishes according to dish size, that is, placing the bigger dishes apart from the smaller dishes. The closest-item-first method may not provide a sorted-size placement of mugs because it only depends on the order of the mugs on the counter-top, which could be random. SA, GA, and PSO methods make a random arrangement of the mugs in order to optimize the placement, which is likely to provide a mixed-size arrangement of the mugs. Hence, the existence of the "sortedness" criterion in the cost function could result in the preference of a sorted-size method, even though it has a lower weight than the other parameters, in the case of the weighted sum cost method. However, for the ranking cost method, this would result in a sorted-size placement to be chosen, only if all the other costs are lower than the mixed-size placement.

In the user study, 46.9% of the participants were women, and 53.1% were men. 63.6% were employed, 62.8% were students, and 89.1% had a dishwasher in their households. According to their replies, 13.2% of the participants use their dishwasher once every two weeks or less, 20.9% use it once a week, 35.7% use it two-three times a week and 30.2% of the participants use it more than three times a week. The highest number of participants were between 18 and 25 years old.

Through the study, we observed that the cost function weights differed slightly depending on the attributes of the participants as presented in Table 5.3. Participants who were students or were employed generally ranked the placement time higher than the electrical energy consumed by the robot. The reason behind the importance of the placement time may be that these types of participants value their time more because of a lack of time in their weekly schedule. The same holds true for those who did not have a dishwasher, or utilized their dishwasher rarely, i.e. once every two weeks or less, or utilized it frequently, i.e. more than three times a week. In fact, the participants who did not have a dishwasher were more concerned with the "sortedness" of the dishes within a tray than with the consumed energy. The placement time was ranked to be more important than the energy for the participants who are older than 51 years old, which is a reasonable result because older people are more concerned with the time.

For the question of approximate placement time of mugs on the counter-top to the dishwasher tray, 56% of the participants replied to place within 3-5 minute range, 31% replied less than 2 minutes, and 13% replied to approximately place the dishes in 6-10 minutes.

5.2.3. Observational study

The observational study that we described in Section 4.2.1.2 was conducted with 21 people, i.e. 10 women and 11 men. In this study, we aimed to examine the packing methods and the heuristics used by people, consciously or subconsciously, when they place dishes into their dishwashers. We also wanted to find the optimum placements for our datasets, such that a ground truth can be set for the maximum number of mugs that can be placed. Furthermore, we wanted to observe if the sortedness parameter was of real importance to people, that is, we wanted to see whether they would place the mugs in a sorted order or not, such that we can compare the results to their responses in the questionnaire. We also conducted an Obsessive-Compulsive Scale test after the placement experiments for each participant, in order to observe whether a relation exists between OCD and the sortedness parameter.

5.2.3.1. Cost Function Weights. The participants filled the questionnaire in Figure C.2, given in Appendix C. The resulting weights in the user preference-based cost function (C_u) for the weighted-sum fitness method are found through the equations presented in Section 4.2.1.5, and they are presented in Table 5.4.

Similar to the user study results, the most important criterion for placing the dishes into the dishwasher is using the maximum capacity of the tray, i.e. maximizing the number of mugs placed. The energy and the placement time have equal weights in the overall results. The results showed that the women participants considered energy to be more important than the placement time, whereas the men considered time as a more important criterion. In general, the weights derived from this study correlate with the weights of the user study. However, the user study is a better ground for deriving generalizations for the categories, as the population size is higher. One important difference between the results of this questionnaire with the results of the user study is that the sortedness has a lower weight. The reason

Table 5.4. The weights of the user-preference based cost function (C_u) using the results of the questionnaire in the annotation study.

Categories	**Attributes**	W_n	W_t	W_E	W_s	**Participants**
Overall Weight		0.38	0.25	0.25	0.11	21 (100.0%)
Gender	Women	0.35	0.24	0.30	0.12	10 (47.6%)
	Men	0.42	0.27	0.21	0.1	11 (52.4%)
Working Status	Working	0.39	0.24	0.26	0.11	13 (62%)
	Not-working	0.37	0.27	0.24	0.12	8 (38%)
Student Status	Student	0.40	0.27	0.24	0.09	18 (86%)
	Not a student	0.3	0.17	0.3	0.23	3 (14%)
Dishwasher Ownership	Owns a dishwasher	0.39	0.24	0.24	0.13	15 (71%)
	Does not have a dishwasher	0.37	0.28	0.28	0.07	6 (29%)
Load Dishwasher	Loads the dishwasher	0.39	0.22	0.24	0.15	12 (57%)
	Does not load	0.39	0.32	0.25	0.04	3 (14%)
Unload Dishwasher	Unloads the dishwasher	0.39	0.22	0.24	0.15	12 (57%)
	Does not unload	0.39	0.32	0.25	0.04	3 (14%)
Usage Frequency	Once every two weeks or less	0.38	0.33	0.23	0.06	5 (24%)
	Once a week	0.33	0.21	0.25	0.22	4 (19%)
	Two-three times a week	0.45	0.19	0.25	0.11	18 (29%)
Loading Time	Less than 2 minutes	0.47	0.19	0.28	0.06	4 (19%)
	3-5 minutes	0.37	0.29	0.24	0.11	14 (67%)
	6-10 minutes	0.37	0.17	0.3	0.17	3 (14%)

behind this fact might be the additional question that we asked in this questionnaire, which was asking for the reason why they think sortedness was important. We observed that this question leads people to think, why they think that the parameter is important. If they could not find a reason, they changed their answer to the previous question to "not important" for the sortedness. The reasons that the participants gave can be categorized under three main subjects, as shown in Table 5.5.

Table 5.5. The reasons for the importance of the sortedness parameter according to the participants of the observational study.

Reason	**Number of responses**
Easiness of unloading the dishwasher and loading the cupboard	7 (50%)
Using the maximum capacity of the dishwasher	6 (43%)
Order within the dishwasher	3 (21.4%)

We initially thought that the reason behind the importance of the sortedness parameter is either the ease of unloading the dishwasher or the likelihood of an OCD for a person. These results confirmed our initial thoughts, but also showed us that the people might have a misjudgment on the advantages of the sortedness on maximizing the capacity of the dishwasher tray.

5.2.3.2. Analysis of the Placements. In our observational study, we asked our participants to place the mugs on the counter-top to the trays for six different datasets as shown in Figure 5.2 and Figure 5.3. We noted the number of mugs placed for each dataset, recorded the amount of time for the placements, and took notes of the placement "patterns" of the participants, that is, their packing methods and the placement order of the items named as the sort methods (κ). The results of these analyses are presented in Table 5.6 and Table 5.7. The "hardest" dataset, that is, the dataset that cannot be placed fully into the dishwasher tray using a trivial sorted placement configuration, is the fourth dataset. It contains 20 big mugs and 10 small mugs. Only 2 participants were successful in placing all of the mugs. The highest

Table 5.6. The number of mugs placed and the placement time results of the observational study and the usage percentages of the sort methods in the placements.

ds	*n*	*t* (min)	**CIF** (%)	**NIS** (%)	**NDS** (%)	**Mixed-Size** (%)
1	12.00±0.00	0.86±0.40	14.29	42.86	19.05	23.81
2	12.00±0.00	0.68±0.30	4.76	52.38	23.81	14.29
3	12.00±0.00	0.63±0.28	4.76	14.29	57.14	14.29
4	28.43±1.21	3.30±2.20	4.76	47.62	4.76	66.67
5	29.95±0.22	1.88±0.59	4.76	38.10	28.57	19.05
6	30.00±0.00	1.61±0.48	4.76	33.33	42.86	14.29

placement time is observed at this dataset because the participants rearranged the items to fit all of them, although they were not requested to do such a task. We asked them to place the mugs, as they would if they were placing them into their own dishwasher. Therefore, it was not required for them to place all of the mugs into the dishwasher, and this statement was made to the participants before and during the experiment. The highest percentages are marked in the table with red, and the second highest scores are marked with blue for comparison. The NIS and NDS denote, placement order according to non-increasing size and non-decreasing size of objects. Mixed-size placements were a result of either placing items initially in a random order or rearranging the mugs within the tray after an initial configuration is made for the tray, which does not contain all the mugs to be placed. We observed that most of our participants placed items with a sort method initially, and if all the items cannot be packed with the sort method, then they rearranged the items and changed their orientations, in order to maximize the capacity of the tray. When the dataset has more small mugs, then NDS is preferred, as was the case with the third and sixth datasets. For the other datasets, most of the participants preferred NIS, even for the datasets with an equal number of small and big mugs, as in the second and fifth datasets. We apply the same approach as our participants in our algorithms, that is, if all the mugs can be placed with an initial sorted configuration, then further optimization is not made. In Table 5.7, we presented the

Table 5.7. The usage percentages of the packing methods in the observational study.

ds	**BLF** (%)	**LBF** (%)	**TLF** (%)	**LTF** (%)	**BRF** (%)	**RBF** (%)	**TRF** (%)	**RTF** (%)	**Mixed** (%)
1	4.76	4.76	23.81	14.29	4.76	0.00	0.00	0.00	52.38
2	4.76	4.76	47.62	0.00	0.00	4.76	4.76	0.00	23.81
3	14.29	4.76	33.33	0.00	4.76	4.76	0.00	0.00	38.10
4	4.76	14.29	23.81	0.00	14.29	4.76	0.00	0.00	38.10
5	9.52	4.76	14.29	9.52	14.29	4.76	0.00	0.00	42.86
6	14.29	14.29	19.05	19.05	4.76	4.76	0.00	0.00	9.52

results of the packing heuristics that we observed from the way the participants place the mugs. BLF, LBF, TLF, LTF, BRF, RBF, TRF, RTF denote Bottom-Left-Fill, Left-Bottom-Fill, Top-Left-Fill, Left-Top-Fill, Bottom-Right-Fill, Right-Bottom-Fill, Top-Right-Fill, and Right-Top-Fill, respectively. The "right" in these methods represent items to be placed in the right-most position instead of the left-most position. In these experiments, the participants preferred mostly the mixed methods that are a combination of these heuristics. TLF was the second best-preferred method.

We observed that the participants learn the patterns that can fit inside the tray after the first dataset that corresponds to a certain tray size and a certain amount of mugs. They place mugs faster after the first dataset that corresponds to the change of tray size and number of mugs. They also use their preferred method of placement for all the datasets, if they can fit all the mugs with that method. If they cannot, then they either change their method or rearrange mugs. Also, the fourth dataset plays an important role in the changes of the packing method and placement order configurations. In other words, if they were unable to fit all the mugs in the fourth dataset, they change their starting placement order and packing method for the fifth dataset such that they can place all of the mugs.

We took a photograph of each placement of the participants in order to have them anno-

tated by other people for their amount of sortedness. The average and medians of sortedness values of each placement and their relation to the results of the OCD test of the participants in the observational study are presented in the following section.

5.2.4. Annotation

For the annotation system described in 4.2.1.3, 63 people (37 women and 26 men) participated in the online anonymous questionnaires given in Appendix D. The results of the annotation consist of three parts: the cost function weights derived from the questionnaire in the annotation, the mean and median sortedness values for the placements in the observational study and the comparison of the OCD state of a participant to the sortedness rank that he/she gives in the questionnaire.

5.2.4.1. Cost Function Weights. The weights of the user-preference based cost function (C_u) are derived from the results of the questionnaire in Figure D.2 in Appendix D. The weights are calculated by the equations in Section 4.2.1.5 and they are presented in Table 5.8.

In general, the weights of each attribute is very similar to the overall weights. However, we found out that some categories of participants value the placement time higher than the energy. These are the categories: men, annotators that are older than 36, working, not a student, does not own a dishwasher, owns a dishwasher but does not do loading or unloading, estimates their own placement time to be greater than 6 minutes, and those who accept robot loading time greater than 6 minutes. In fact, those who are between 36 to 50 years old, and estimate their own placement time to be greater than 10 minutes give more importance to time than maximizing the capacity, which might mean that these people have limited time in their daily lives. The sortedness weight has decreased compared to the previous questionnaires, which might be as a result of the question asking for the reason behind the sortedness, as described in the observational study. We decided to use the overall weights found in the annotation as the weights of the cost function, because of that reason.

Table 5.8. The weights of the user-preference based cost function (C_u) using the results of the questionnaire in the annotation study.

Categories	**Attributes**	W_n	W_t	W_E	W_s	**Annotators**
Overall Weight		0.37	0.27	0.28	0.08	63 (100.0%)
Gender	Women	0.36	0.26	0.29	0.09	37 (59.7%)
	Men	0.37	0.28	0.27	0.08	26 (41.3%)
Age Range	18-25	0.36	0.27	0.28	0.09	35 (56.5%)
	26-35	0.39	0.24	0.28	0.08	18 (28.6%)
	36-50	0.31	0.33	0.30	0.06	7 (12.1%)
	51-65	0.43	0.29	0.25	0.04	3 (7.9%)
Working Status	Working	0.37	0.28	0.27	0.09	40 (63.5%)
	Not-working	0.36	0.26	0.30	0.08	23 (37.1%)
Student Status	Student	0.37	0.25	0.29	0.08	39 (61.9%)
	Not a student	0.36	0.29	0.27	0.08	24 (39.3%)
Dishwasher Ownership	Owns a dishwasher	0.36	0.26	0.28	0.09	53 (85.5%)
	Does not have a dishwasher	0.37	0.33	0.27	0.03	10 (15.9%)
Load Dishwasher	Loads the dishwasher	0.37	0.25	0.29	0.09	45 (72.6%)
	Does not load	0.32	0.31	0.27	0.11	8 (13.3%)
Unload Dishwasher	Unloads the dishwasher	0.37	0.25	0.29	0.09	46 (74.2%)
	Does not unload	0.31	0.34	0.26	0.09	7 (11.7%)
Usage Frequency	Once every two weeks or less	0.42	0.26	0.26	0.05	2 (3.5%)
	Once a week	0.34	0.26	0.27	0.13	17 (27.9%)
	Two-three times a week	0.40	0.25	0.28	0.06	18 (30.5%)
	More than three times a week	0.35	0.26	0.30	0.09	16 (25.8%)
Loading Time	Less than 2 minutes	0.37	0.23	0.30	0.10	19 (30.2%)
	3-5 minutes	0.37	0.27	0.28	0.08	36 (59.0%)
	6-10 minutes	0.38	0.32	0.23	0.08	6 (10.2%)
	More than 10 minutes	0.26	0.42	0.26	0.05	2 (6.7%)
Preferred Robot Loading Time	Less than 2 minutes	0.37	0.28	0.28	0.08	31 (50.8%)
	3-5 minutes	0.35	0.24	0.31	0.09	19 (30.2%)
	6-10 minutes	0.35	0.28	0.27	0.11	10 (19.2%)
	10-15 minutes	0.62	0.23	0.15	0.00	2 (6.3%)
	More than 15 minutes	0.44	0.33	0.22	0.00	1 (3.6%)

Another attribute that we examined in this questionnaire was the maximum preferred loading time range of the annotators. According to the replies of the annotators, 50.8% prefer a maximum of two minutes, 30.2% prefer the robot to finish the placement within three to five minutes, 19.2% prefer the placement to be up to 10 minutes, 6.3% prefers a maximum of 15 minutes and 3.6%, that is, one person would allow the robot to finish its work in more than 15 minutes.

According to different attributes of the annotators, the weights of the engineering cost function (C_e), defined in Eq. 4.10 in Section 4.2.1.5, are presented in Table 5.9. The overall weights found in Table 5.9 are used for the engineering cost function. The analysis made for Table 5.8 is also valid for this table. The weights of the sortedness parameter are shared among the other parameters according to the number of responses, which enable us to observe the differences between the energy and the time weights more clearly.

The general reasons behind the importance of the sortedness parameter are presented in Table 5.10. As denoted before in the observational study, the biggest reason behind the sortedness parameter is the concern for the ease of unloading the dishwasher. The additional reasons that are found in this questionnaire are the cleanliness, and preventing collisions. The state of getting mugs cleaned does not depend on the sortedness of the order, therefore, this reason is not considered valid in our case. Furthermore, we use tolerance between the mugs in order to avoid any collisions between the dishes.

5.2.4.2. Sortedness Annotation with Likert Scale. We used a 5-point Likert scale in order to rate the sortedness of the placements from the observational study. However, since an annotator can have a different interpretation of the sortedness or order of the items according to size, and may not understand what is requested, we decided to use three training images in order to let the annotator see the correspondence of the placements to the scale. This allowed us to train our annotators without biasing them. The three training images that correspond to the “very ordered”, “very unordered” and “moderately-ordered” cases are presented in Figure 5.5.

Table 5.9. The weights of the engineering cost function (C_e) using the results of the questionnaire in the annotation study.

Categories	**Attributes**	W_n	W_t	W_E	**Annotators**
Overall Weight		0.43	0.27	0.30	63 (100.0%)
Gender	Women	0.43	0.26	0.31	37 (59.7%)
	Men	0.43	0.29	0.27	26 (41.3%)
Age Range	18-25	0.42	0.28	0.30	35 (56.5%)
	26-35	0.47	0.23	0.30	18 (28.6%)
	36-50	0.33	0.36	0.31	7 (12.1%)
	51-65	0.50	0.28	0.22	3 (7.9%)
Working Status	Working	0.43	0.29	0.28	40 (63.5%)
	Not-working	0.42	0.25	0.33	23 (37.1%)
Student Status	Student	0.44	0.25	0.31	39 (61.9%)
	Not a student	0.42	0.31	0.27	24 (39.3%)
Dishwasher Ownership	Owns a dishwasher	0.44	0.26	0.30	53 (85.5%)
	Does not have a dishwasher	0.40	0.34	0.26	10 (15.9%)
Load Dishwasher	Loads the dishwasher	0.45	0.24	0.31	45 (72.6%)
	Does not load	0.37	0.35	0.28	8 (13.3%)
Unload Dishwasher	Unloads the dishwasher	0.45	0.24	0.31	46 (74.2%)
	Does not unload	0.34	0.39	0.26	7 (11.7%)
Usage Frequency	Once every two weeks or less	0.50	0.25	0.25	2 (3.5%)
	Once a week	0.42	0.27	0.31	17 (27.9%)
	Two-three times a week	0.47	0.24	0.28	18 (30.5%)
	More than three times a week	0.41	0.26	0.33	16 (25.8%)
Loading Time	Less than 2 minutes	0.44	0.23	0.33	19 (30.2%)
	3-5 minutes	0.43	0.27	0.29	36 (59.0%)
	6-10 minutes	0.44	0.34	0.22	6 (10.2%)
	More than 10 minutes	0.25	0.50	0.25	2 (6.7%)
Preferred Robot Loading Time	Less than 2 minutes	0.43	0.29	0.28	31 (50.8%)
	3-5 minutes	0.42	0.24	0.35	19 (30.2%)
	6-10 minutes	0.43	0.30	0.28	10 (19.2%)
	10-15 minutes	0.67	0.22	0.11	2 (3.6%)
	More than 15 minutes	0.50	0.33	0.17	1 (6.3%)

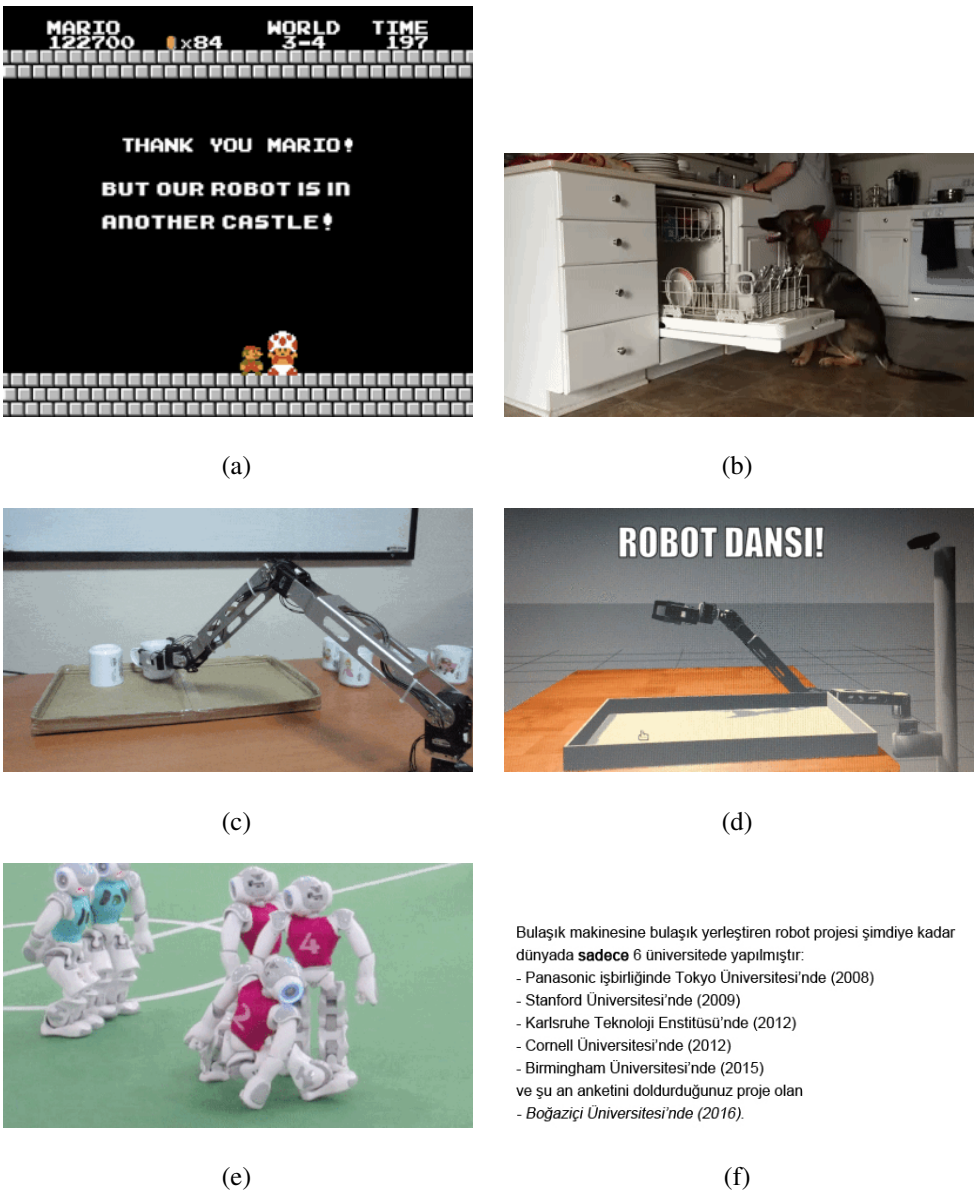
Table 5.10. The reasons for the importance of the sortedness parameter according to the annotators.

Reason	**Number of responses**
Easiness of unloading the dishwasher and loading the cupboard	14 (56%)
Using the maximum capacity of the dishwasher	5 (20%)
Order within the dishwasher	5 (20%)
Cleanliness	2 (8%)
Preventing collision between the dishes and with the dishwasher compartments	2 (8%)

(a) (b) (c)

Figure 5.5. Training placement images in the annotation that correspond to: (a) Very ordered placement, (b) Very unordered placement, (c) Moderately-ordered.

After these three images, 42 placement images from the observational study, that correspond to all the placements in two of the datasets, are presented in a random order one-by-one to the annotator. In order to prevent annotators from losing their concentration and providing random results, we used animated images related to robots and dishwashers after each 7 placement images. The animated images last 5 seconds, such that annotators can rest at least 5 seconds before they continue. These images are presented in Figure 5.6.



(a) (b)

(c) (d)

(e) (f)

Figure 5.6. Animated images related to robots and dishwashers.

The mean and the standard deviations of the highest, lowest, and average sortedness

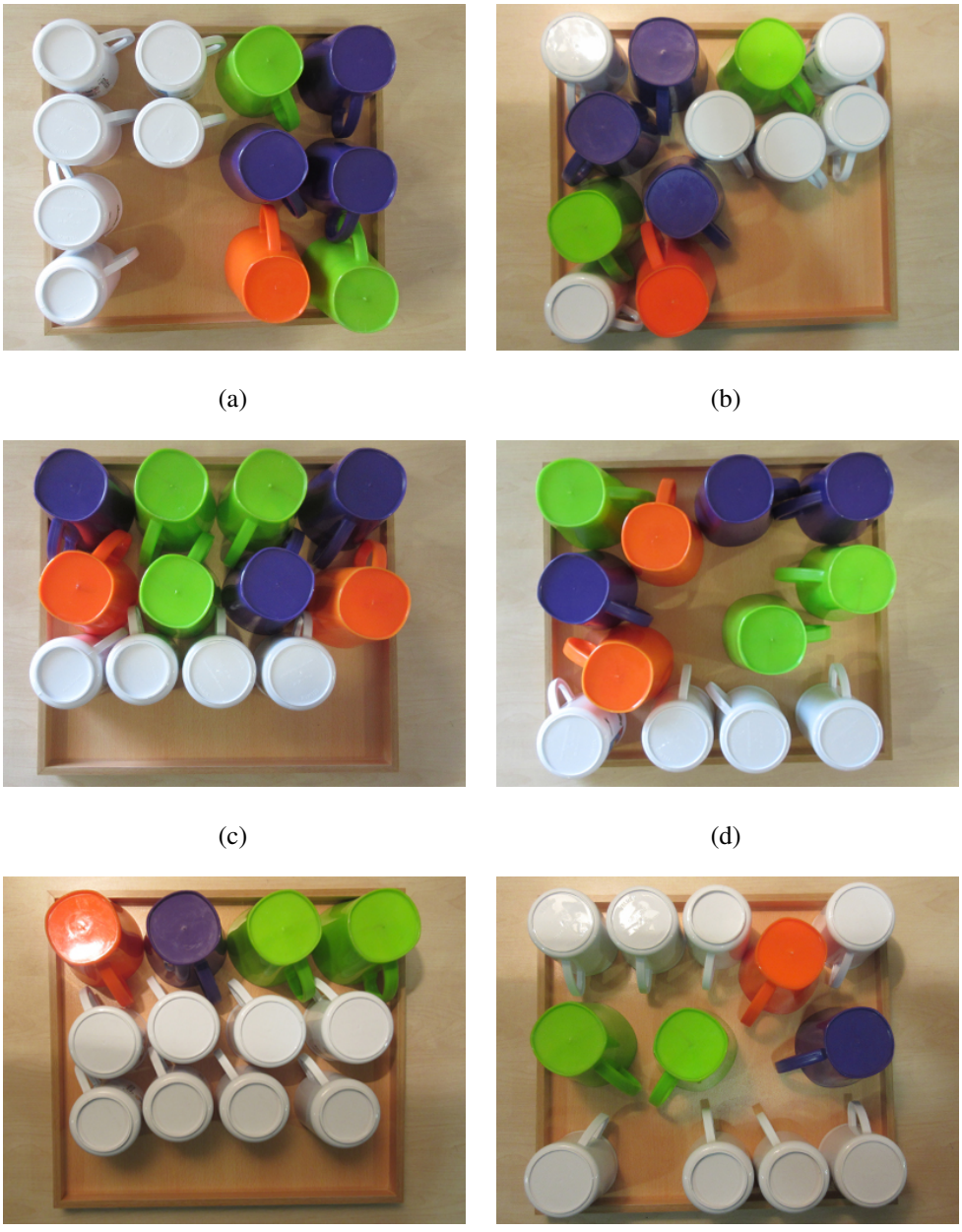
in the datasets are presented in Table 5.11, and the median and the interquartile range of the highest, lowest, and average sortedness in the datasets are presented in Table 5.12.

Table 5.11. The mean and standard deviation of the highest, lowest, and mean of the sortedness in the datasets of the observational study.

ds	**Highest Sortedness**	**Lowest Sortedness**	**Mean Sortedness in Dataset**
Training	4±0.82	2.5±0.99	3.25±0.96
1	3.7±0.95	1.95±0.92	3.36±0.51
2	3.95±0.93	2.23±1.08	3.18±0.51
3	4.38±0.72	2.14±0.89	3.54±0.62
4	4.3±0.64	1.95±1.24	3.14±0.66
5	4.31±0.63	1.95±1.14	3.13±0.62
6	4.52±0.5	1.57±0.79	3.49±0.71

According to the mean and standard deviation table, we can observe that the training images have achieved almost correct mean values that we wanted, which indicate that we are successful in training our annotators. In all the datasets, the mean values are around 3, which means that in general, people place in a moderately-sorted way. The median values show the same results, except for the sixth dataset. The third and sixth datasets have higher sortedness values because they are "easier" than the other datasets, that is, they have more small mugs than the big mugs, which allows fitting all the items in a sorted order for our datasets. The Figure 5.7 and 5.8 show the placements with the highest and lowest sorted placements according to the average sortedness values. The median values are not used for finding the highest lowest values because several images have the same median values which prevent comparisons between them to find the highest or the lowest cases.

By examining these images, we can conclude that our annotation system was successful in rating the sortedness of each placement, as our idea of a sorted placement is rated as sorted in the overall, and the unsorted placement is rated as unsorted. Hence, we can use

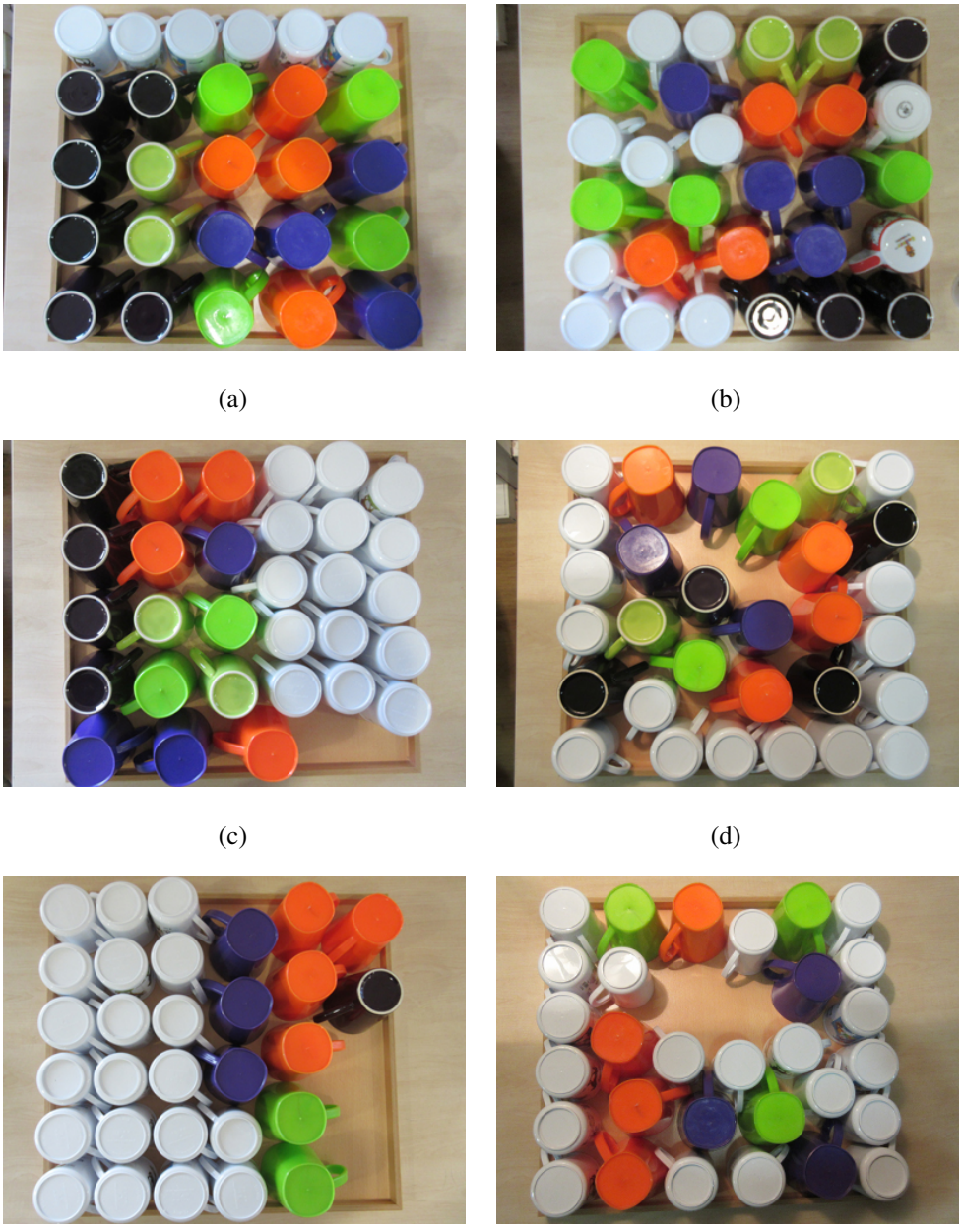


(a) (b)

(c) (d)

(e) (f)

Figure 5.7. The placements that correspond to the highest, and the lowest sortedness in the datasets of the observational study: (a) Highest sortedness in dataset 1, (b) Lowest sortedness in dataset 1, (c) Highest sortedness in dataset 2, (d) Lowest sortedness in dataset 2, (e) Highest sortedness in dataset 3, (f) Lowest sortedness in dataset 3.



(a) (b)

(c) (d)

(e) (f)

Figure 5.8. The placements that correspond to the highest, and the lowest sortedness in the datasets of the observational study: (a) Highest sortedness in dataset 4, (b) Lowest sortedness in dataset 4, (c) Highest sortedness in dataset 5, (d) Lowest sortedness in dataset 5, (e) Highest sortedness in dataset 6, (f) Lowest sortedness in dataset 6.

Table 5.12. The median and interquartile range of the highest, lowest, and median of the sortedness in the datasets of the observational study.

ds	**Highest Sortedness**	**Lowest Sortedness**	**Median Sortedness in Dataset**
Training	4±2	2±1	3±2
1	4±1	3±1	3±0
2	4±1	3±1	3±0
3	4±1	3±1	3±1
4	4±1	3±1	3±0
5	4±1	3±1	3±0
6	4±1	3±1	4±1

these sortedness values to determine whether a participant in the observational study that ranked the sortedness parameter as important, i.e. ranked with a value greater than zero, has actually placed in a sorted manner, or not. Our aim behind the annotation study was to observe if the sortedness parameter actually matters for people or not. We categorized the average results of each participant as "sorted" for mean sortedness values greater than 3 and "not-sorted" for mean sortedness less than 3. The results of the comparison is given in Table 5.13.

We can observe that more than 50% of the participants who ranked the sortedness parameter as important, have placed items in a sorted way. The same is true for the participants that ranked the sortedness as unimportant, except for the fourth dataset, which restricted the use of a sorted placement in order to increase the number of mugs placed. However, our approach for categorizing the sortedness values into "sorted" and "not-sorted" placements does not consider the moderately-sorted case, in which a placement is neither "sorted", nor "not-sorted". Therefore, we examined the median values of each placement, and categorized values above 3 as "sorted" and below 3 as "not-sorted". In other words, if the median for a placement is 1 or 2, then it is "not-sorted", and if it is 4 or 5, then it is "sorted". The percentages of the participants for each dataset using the median and inter-quartile range for

Table 5.13. The comparison of the sortedness ranks of participants in the questionnaire to the sortedness categories of their placements in each dataset using mean and standard deviation for the Likert scale.

ds	**Rank Not Zero**		**Rank Zero**	
	Sorted (%)	**Not-Sorted** (%)	**Sorted** (%)	**Not-Sorted** (%)
1	93	7	71	29
2	57	43	86	0
3	71	29	86	14
4	64	36	43	57
5	64	36	57	43
6	71	29	86	14

sortedness are given in Table 5.14.

These results show that in each set except the sixth dataset, more than 50% of the participants who ranked the sortedness as important placed the mugs in a moderately-sorted way. As noted before, the percentage of participants that make a sorted placement decreases with increasing "hardness" of the problem, that is, less amount of sorted placements occur, when the solution for the packing problem is not trivial and cannot be solved by a sorted placement. All of the participants who ranked the sortedness as unimportant placed in a moderately-sorted way as well. Therefore, we can conclude that the sortedness parameter is not very important in the placement, hence, it can be ignored for the weighted-sum method. The fitness ranking method compares the sortedness parameter as the last comparison, therefore, it is not affected by this conclusion.

5.2.4.3. Obsessive-Compulsive Scale Test. We conducted the Obsessive-Compulsive Scale test through the questionnaire given in Figure C.3 in Appendix C in the observational study and the same test as given in Figure D.4 in Appendix D in the annotation system. We analyzed the effect of likeliness of the Obsessive-Compulsive Disease (OCD) in a person

Table 5.14. The comparison of the sortedness ranks of participants in the questionnaire to the sortedness categories of their placements in each dataset using median and inter-quartile range for the Likert scale.

ds	**Rank Not Zero**		**Rank Zero**	
	Sorted (%)	**Not-Sorted** (%)	**Sorted** (%)	**Not-Sorted** (%)
1	24	0	0	0
2	10	0	0	0
3	43	0	0	0
4	19	0	0	0
5	14	0	0	0
6	62	0	0	0

on ranking the sortedness parameter as important. The results are given in Table 5.15.

The results show that there is a relation between the importance of the sortedness parameter and the OCD state. The importance of the sortedness parameter decreases, a person is less likely to have OCD. We further examine the results of the participants by their sorted placements, and find the percentages of each category using mean sortedness values in Table 5.16 and using median sortedness in Table 5.17. The percentages correspond to the number of people that ranked the sortedness parameter as important or unimportant and placed in a “sorted” or “not-sorted” divided by the number of participants at an OCD state.

The results of the comparison of OCD states to ranks and sorted placements show that the state of the OCD indeed affects both the rank and the sorted placement of the people. However, OCD is not the only reason behind sortedness, as there are other reasons as shown in Table 5.10.

Table 5.15. The comparison of Obsessive-Compulsive Scale of the participants and annotators to the sortedness ranks within the questionnaire.

OCD State	**Rank Not Zero** (%)	**Rank Zero** (%)	**Number of People**
Likely Annotator	41	59	27
Likely Participant	83	17	12
Probable Annotator	50	50	22
Probable Participant	43	57	7
Unlikely Annotator	29	71	14
Unlikely Participant	50	50	2

Table 5.16. The comparison of the Obsessive-Compulsive Scale of the participants to their sortedness ranks and the sortedness category according to their overall placements using mean of values.

OCD State	**Rank Not Zero**		**Rank Zero**		**Number of Participants**
	Sorted (%)	**Not-Sorted** (%)	**Sorted** (%)	**Not-Sorted** (%)	
Likely Participant	67	17	8	8	12
Probable Participant	43	0	57	0	7
Unlikely Participant	50	0	50	0	2

Table 5.17. The comparison of the Obsessive-Compulsive Scale of the participants to their sortedness ranks and the sortedness category according to their overall placements using median of values.

OCD State	**Rank Not Zero**		**Rank Zero**		**Number of Participants**
	Sorted (%)	**Not-Sorted** (%)	**Sorted** (%)	**Not-Sorted** (%)	
Likely Participant	33	0	0	0	12
Probable Participant	0	0	0	0	7
Unlikely Participant	0	0	0	0	2

5.2.5. Weighted Sum Fitness Method

The cost functions for the weighted-sum method are found through the overall weights from the first questionnaire of the annotation from Table 5.8 for user preference-based cost and from Table 5.9 for the engineering cost.

The bin-packing cost function is:

$$C_{b_{WS}} = (1 - n') \tag{5.1}$$

The user preference based cost function is:

$$C_{u_{WS}} = 0.37(1 - n') + 0.28E' + 0.27t' + 0.08(1 - s) \tag{5.2}$$

and the engineering cost is:

$$C_{e_{WS}} = 0.43(1 - n') + 0.3E' + 0.27t' \tag{5.3}$$

5.2.6. Ranking Fitness Method

According to the weights used in the weighted-sum method, we can conclude that the most important criterion is using the maximum capacity of the dishwasher, which can be achieved by using the maximum number of mugs that cover the maximum area within the tray. Therefore, the bin-packing cost of the ranking fitness method is defined as:

$$C_{b_{RF}} = \begin{cases} 0 & \text{if } n'_c > n'_p \text{ or } \left(n'_c = n'_p \text{ and } A'_c > A'_p\right) \\ 1 & \text{otherwise} \end{cases} \tag{5.4}$$

The second most important criterion is the energy consumption, followed by the placement time and sortedness of a placement. Therefore, the user preference based cost function is defined as follows:

$$C_{u_{RF}} = \begin{cases} 0 & \text{if} n'_c > n'_p \\ & \text{or } \left(n'_c = n'_p \text{ and } A'_c > A'_p\right) \\ & \text{or } \left(n'_c = n'_p \text{ and } A'_c = A'_p \text{ and } E'_c < E'_p\right) \\ & \text{or } \left(n'_c = n'_p \text{ and } A'_c = A'_p \text{ and } E'_c = E'_p \text{ and } t'_c < t'_p\right) \\ & \text{or } \left(n'_c = n'_p \text{ and } A'_c = A'_p \text{ and } E'_c = E'_p \text{ and } t'_c = t'_p \text{ and } s_c > s_p\right) \\ 1 & \text{otherwise} \end{cases} \tag{5.5}$$

The engineering cost function omits the sortedness comparison above.

5.3. Placement Tests

We tested different placement methods for our settings: non-decreasing width, non-increasing width, non-decreasing length, non-increasing width, closest-mug-first and SA, GA, PSOp, and PSOk optimization methods. We conducted 21 runs for each case and recorded the number of mugs placed in each run and the total time it took for the placement. Our configuration space and mug sizes for the robot experiments only allowed for 12 mugs to be used in the dataset. The robot is accurately represented in the *Gazebo* simulation environment, therefore, it is not possible to simulate a dataset that is out of reach for the robot

arm, as the higher number of mugs means that the mugs will be outside of the configuration space of the robot. Therefore, for testing the 30 mug datasets with the bigger tray, we could not use the *Gazebo* environment. Hence, we tested our results in the C++ programming language. In order to calculate the costs which would model the real world for a bigger tray setting, we used the counter-top mug coordinates from our observational study. We calculated the results on a computer with 16 GB RAM, Intel Core i7-2600 CPU at 3.40 GHz x 8, running Ubuntu 14.04 OS. We use *MATLAB R2014a* in order to visualize the results of the placement within the tray. We conducted a tolerance test to find the minimum tolerance amount between each mug that would ensure a placement without collisions, and the results of the test are given in Section 5.3.1. The methods and heuristics used in placement planning as described in Section 4.2 are summarized in Table 5.18. The best results of the simulation tests and their analysis are presented in Section 5.3.2.

Table 5.18. Summary of the methods and the heuristics used in placement planning.

Optimization Method	**Packing Heuristic**	**Sort Method**	**Fitness Method**	**Fitness Function**
Simulated Annealing (SA)	Left-Top-Fill (LTF)	Closest-Item-First (CIF)	Weighted-Sum (WS)	Bin-Packing Cost (BPC)
Genetic Algorithm (GA)	Top-Left-Fill (TLF)	Non-increasing Size (NIS)	Ranking Fitness (RF)	User Preference-Based Cost (UPC)
Particle Swarm Optimization with Key (PSOk)		Non-decreasing Size (NDS)		Engineering Cost (EC)
Particle Swarm Optimization with Position (PSOp)		Random Order (RO)		

For the Simulated Annealing algorithm, the temperature (T), the cooling rate (cr) and the absolute temperature (T_{abs}) (the stopping temperature) are empirically determined to be $T = 1000$, $cr = 0.99$ and $T_{abs} = 0.01$ through the simulation tests in C++, which resulted in low costs and low calculation times. For the Genetic Algorithm, the parameter values are used from Gehring and Bortfeldt [43]. The size of the population ($npop$), the number of generations ($ngen$) and the number of reproduced individuals ($nrep$), the probability of a crossover (PC) and the probability of a mutation (PM) are $npop = 50$, $ngen = 30$, $nrep =$

10, $PC = 0.67$ and $PM = 0.33$, respectively. *ngen* is decreased empirically, to find lower calculation times for the algorithm. Finally, for both of the Particle Swarm Optimization methods, maximum number of iterations (τ_{max}) is 500, and the probability of a improved PSO (*PS*) is 0.5, which are determined empirically. The minimum inertia (ω_{min}) is 0.4, the maximum inertia (ω_{max}) is 0.9, and the acceleration coefficients are $c_1 = 1.5$, $c_2 = 1.5$ and $c_3 = 5.0$ as in [52].

5.3.1. Tolerance Test

In order to optimize the tolerance amount between mugs in the tray, we conducted tests on the placement with different types of mugs, with increasing tolerance amount in width and length. The number of mugs placed in the tray and the existence of any mug collisions depending on the tolerance amount is presented in Table 5.19. The collisions occurred between

Table 5.19. Tolerance test results with 11 mugs.

Mug Type	**Tolerance** (cm)	**Calculated Number of Mugs to Place**	**Mug Collisions**
Small Mugs	0.0	11	3
Large Mugs		10	2
Small Mugs	0.5	11	1
Large Mugs		11	2
Small Mugs	1.0	11	0
Large Mugs		9	0
Small Mugs	1.5	11	0
Large Mugs		9	0

the gripper and the mugs, and the gripper could not descend to the necessary position. The reason for the collisions was the position error in the arm. Although at 0.5 cm, the number of collisions is low and the number of mugs placed is relatively higher, in a real life setting the mugs will be fragile objects, therefore, there should not be any collisions with the mugs.

Hence, we have set our tolerance limit at 1.0 cm.

5.3.2. Simulation Tests

The tables for the mean and standard deviation values for each cost parameter for 21 iterations of the combinations of optimization methods and heuristics are available online [110]. In these tables, the methods that result in the lowest cost are marked with red. The LTF and TLF tables are separated, however, the lowest cost is found for the overall, that is, only one lowest cost is found for a parameter. In other words, if a table does not have a red marked cell, it means that the packing method (LTF or TLF) was unsuccessful in providing the lowest cost for that parameter. In order to compare the costs of each method, we normalized the values of each parameter according to the maximum and minimum values in a dataset, and applied weighted sum and fitness ranking methods to find the actual costs of the methods. These resulting cost tables can be found in the link given as well. We observed from these tables that there are multiple best solutions. Therefore, we chose the solution out of the lowest cost solutions that has the minimum execution time, and closest-item-first method is preferred for minimizing the energy consumption. The best methods that result in the lowest cost for each of the fitness method and fitness function combinations are presented in Tables 5.20 - 5.23. The cost for the weighted-sum method and the rank for the ranking fitness method are given in each cell for the corresponding tables below the method combination name, followed by the execution times of the algorithms in minutes.

The results in Tables 5.20 - 5.23 show that for our datasets, in general, SA with the LTF packing heuristic provides the best results for the ranking fitness functions and GA with the LTF packing heuristic provides the best results for the weighted-sum cost functions. PSOk_i provides better results than PSOk_n and PSOp_n gives better results than PSOp_i. Analyzing the cost values, we can say that the PSOp methods are not suitable for using in our problem, as they provide the highest costs, with the lowest number of mugs placed. BH heuristics can provide the lowest cost depending on the dataset. For example, it could not find the best placement in any of the fitness methods and functions in the second dataset. Also, the ranking fitness considers the covered area, therefore, even though BH might produce the lowest costs in a weighted-sum method, it might not be the best solution for the fitness ranking method.

Table 5.20. The heuristic combinations for the optimization methods that result in lowest bin-packing costs with weighted-sum method.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	TLF:NIS:RF:BPC 0.00 : 0.1466	LTF:NDS:RF:BPC 0.00 : 0.1926	LTF:NIS:WS:BPC 0.81 : 0.5481	LTF:CIF:RF:BPC 0.84 : 0.6699	LTF:NDS:RF:EC 0.00 : 0.2307	LTF:CIF:RF:EC 0.00 : 0.2309	LTF:NDS:RF:BPC 0.00 : 0.0001	BH:LTF:NDS:RF:BPC 0.00 : 0.0001
2	TLF:CIF:WS:EC 0.00 : 0.1351	LTF:RO:RF:EC 0.00 : 0.1644	LTF:RO:WS:BPC 0.67 : 0.6004	TLF:RO:RF:BPC 0.76 : 0.6490	LTF:CIF:RF:BPC 0.00 : 0.2266	LTF:RO:RF:BPC 0.00 : 0.2301	TLF:RO:RF:UPC 0.32 : 0.0001	SA:TLF:CIF:WS:EC 0.00 : 0.1351
3	TLF:NDS:WS:UPC 0.00 : 0.1633	LTF:CIF:RF:EC 0.00 : 0.1815	LTF:CIF:RF:BPC 0.73 : 0.6266	LTF:NIS:RF:EC 0.78 : 0.6850	LTF:CIF:WS:BPC 0.00 : 0.2384	LTF:RO:WS:BPC 0.00 : 0.2314	LTF:NDS:WS:BPC 0.00 : 0.0001	BH:LTF:NDS:WS:BPC 0.00 : 0.0001
4	TLF:RO:RF:EC 0.00 : 0.4558	TLF:NDS:RF:BPC 0.00 : 0.7174	LTF:RO:RF:BPC 0.88 : 1.1042	LTF:RO:RF:BPC 0.91 : 1.1060	TLF:NDS:RF:EC 0.00 : 0.8682	TLF:NIS:WS:BPC 0.00 : 0.6672	LTF:NDS:RF:BPC 0.00 : 0.0003	BH:LTF:NDS:RF:BPC 0.00 : 0.0003
5	TLF:CIF:RF:EC 0.00 : 0.5152	TLF:NDS:RF:BPC 0.00 : 0.7239	LTF:RO:WS:BPC 0.87 : 1.0836	LTF:NDS:WS:BPC 0.90 : 1.0841	TLF:NDS:RF:BPC 0.00 : 0.8906	TLF:NDS:RF:EC 0.00 : 0.6944	LTF:NDS:RF:EC 0.00 : 0.0002	BH:LTF:NDS:RF:EC 0.00 : 0.0002
6	TLF:NDS:RF:UPC 0.00 : 0.5162	LTF:NDS:RF:EC 0.00 : 0.6635	LTF:RO:RF:BPC 0.84 : 1.2837	LTF:RO:RF:EC 0.88 : 1.2448	LTF:NDS:RF:BPC 0.00 : 0.6146	LTF:NDS:WS:BPC 0.00 : 0.6221	LTF:NDS:WS:BPC 0.00 : 0.0003	BH:LTF:NDS:WS:BPC 0.00 : 0.0003

Table 5.21. The heuristic combinations for the optimization methods that result in lowest bin-packing costs with ranking fitness method.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	TLF:NIS:RF:BPC 1 : 0.1466	LTF:NDS:RF:BPC 1 : 0.1926	LTF:NIS:RF:BPC 76 : 0.5184	LTF:CIF:RF:BPC 77 : 0.6699	LTF:NDS:RF:EC 1 : 0.2307	LTF:CIF:RF:EC 1 : 0.2309	LTF:NDS:RF:BPC 1 : 0.0001	BH : LTF:NDS:RF:BPC 1 : 0.0001
2	TLF:CIF:WS:EC 1 : 0.1351	LTF:RO:RF:EC 1 : 0.1644	LTF:RO:WS:BPC 44 : 0.6004	LTF:RO:RF:BPC 60 : 0.6490	LTF:CIF:RF:BPC 1 : 0.2266	LTF:RO:RF:BPC 1 : 0.2301	TLF:RO:RF:UPC 24 : 0.0001	SA : TLF:CIF:WS:EC 1 : 0.1351
3	LTF:CIF:RF:BPC 1 : 0.2532	LTF:CIF:RF:EC 2 : 0.1815	LTF:NDS:WS:BPC 63 : 0.5997	LTF:NIS:RF:EC 63 : 0.6850	LTF:CIF:RF:BPC 1 : 0.2442	LTF:CIF:RF:BPC 1 : 0.2349	LTF:NIS:WS:BPC 2 : 0.0001	PSOk$_i$: LTF:CIF:RF:BPC 1 : 0.2349
4	LTF:RO:RF:UPC 1 : 0.6532	TLF:NDS:RF:BPC 11 : 0.7174	LTF:CIF:RF:BPC 152 : 1.0383	LTF:RO:RF:EC 156 : 1.0621	TLF:NDS:RF:BPC 9 : 0.8788	TLF:NIS:RF:BPC 1 : 0.6919	LTF:NDS:RF:BPC 23 : 0.0003	SA : LTF:RO:RF:UPC 1 : 0.6532
5	LTF:RO:RF:BPC 1 : 0.6666	TLF:NDS:RF:BPC 13 : 0.7239	LTF:RO:WS:BPC 171 : 1.0836	LTF:NDS:WS:BPC 185 : 1.0841	TLF:NIS:WS:BPC 8 : 0.9176	TLF:NDS:RF:BPC 9 : 0.7330	LTF:NDS:RF:EC 22 : 0.0002	SA : LTF:RO:RF:BPC 1 : 0.6666
6	TLF:NDS:RF:UPC 1 : 0.5162	LTF:NDS:RF:EC 1 : 0.6635	LTF:RO:RF:BPC 153 : 1.2837	LTF:RO:RF:EC 163 : 1.2448	LTF:NDS:RF:BPC 1 : 0.6146	LTF:NDS:WS:BPC 1 : 0.6221	LTF:NDS:WS:BPC 1 : 0.0003	BH : LTF:NDS:WS:BPC 1 : 0.0003

Table 5.22. The heuristic combinations for the optimization methods that result in lowest user preference-based costs with weighted-sum.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	TLF:NDS:WS:EC 0.43 : 0.1464	TLF:RO:WS:UPC 0.42 : 0.2166	LTF:NDS:RF:UPC 0.50 : 0.5748	LTF:NDS:RF:EC 0.50 : 0.5722	LTF:NDS:WS:UPC 0.43 : 0.2390	LTF:NDS:WS:UPC 0.42 : 0.2617	TLF:NDS:RF:EC 0.46 : 0.0001	GA:TLF:RO:WS:UPC 0.42 : 0.2166
2	TLF:NIS:RF:EC 0.33 : 0.1511	TLF:RO:WS:UPC 0.37 : 0.2083	LTF:NDS:RF:UPC 0.52 : 0.7081	LTF:NDS:WS:EC 0.53 : 0.6828	LTF:NDS:WS:UPC 0.37 : 0.2484	LTF:NDS:WS:UPC 0.35 : 0.2566	TLF:NDS:RF:EC 0.46 : 0.0000	SA:TLF:NIS:RF:EC 0.33 : 0.1511
3	TLF:NDS:WS:UPC 0.40 : 0.1633	LTF:NDS:WS:UPC 0.38 : 0.2101	LTF:NDS:WS:BPC 0.49 : 0.5997	LTF:NDS:WS:BPC 0.50 : 0.6767	LTF:RO:WS:UPC 0.41 : 0.2654	LTF:RO:WS:UPC 0.39 : 0.2774	TLF:NDS:RF:EC 0.44 : 0.0001	GA:LTF:NDS:WS:UPC 0.38 : 0.2101
4	TLF:NDS:WS:UPC 0.43 : 0.4577	LTF:RO:WS:UPC 0.38 : 0.6192	LTF:NDS:RF:BPC 0.42 : 1.0742	LTF:NDS:RF:UPC 0.44 : 1.7872	LTF:CIF:WS:UPC 0.45 : 0.6494	LTF:CIF:WS:UPC 0.44 : 0.7098	TLF:NDS:RF:EC 0.44 : 0.0002	GA:LTF:RO:WS:UPC 0.38 : 0.6192
5	LTF:NDS:WS:UPC 0.45 : 0.6853	LTF:NIS:WS:UPC 0.38 : 0.6082	LTF:NIS:WS:EC 0.47 : 1.0589	LTF:NIS:RF:BPC 0.48 : 1.0739	TLF:NIS:WS:UPC 0.44 : 0.9218	LTF:NIS:WS:UPC 0.44 : 0.7218	LTF:NIS:RF:BPC 0.42 : 0.0002	GA:LTF:NIS:WS:UPC 0.38 : 0.6082
6	LTF:NDS:WS:EC 0.46 : 0.6552	LTF:NDS:WS:UPC 0.44 : 0.5428	LTF:CIF:RF:UPC 0.48 : 1.4197	LTF:CIF:WS:UPC 0.48 : 1.3720	LTF:NDS:WS:UPC 0.50 : 0.6782	LTF:NDS:WS:UPC 0.50 : 0.7070	TLF:NIS:RF:EC 0.50 : 0.0002	GA:LTF:NDS:WS:UPC 0.44 : 0.5428

Table 5.23. The heuristic combinations for the optimization methods that result in lowest user preference-based costs with ranking fitness.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	LTF:CIF:RF:UPC 7 : 0.2400	LTF:NDS:RF:BPC 31 : 0.1926	LTF:NIS:RF:BPC 210 : 0.5184	LTF:CIF:RF:BPC 211 : 0.6699	LTF:CIF:RF:UPC 5 : 0.2847	LTF:NIS:RF:EC 1 : 0.2469	LTF:NDS:WS:BPC 32 : 0.0001	PSOk$_i$: LTF:NIS:RF:EC 1 : 0.2469
2	TLF:NIS:RF:EC 26 : 0.1511	LTF:CIF:RF:UPC 35 : 0.1818	LTF:RO:WS:BPC 206 : 0.6004	LTF:RO:RF:BPC 228 : 0.6490	LTF:NDS:RF:UPC 11 : 0.2460	LTF:RO:RF:UPC 1 : 0.2656	TLF:RO:RF:UPC 184 : 0.0001	PSOk$_i$: LTF:RO:RF:UPC 1 : 0.2656
3	LTF:CIF:RF:UPC 1 : 0.2544	LTF:CIF:RF:EC 6 : 0.1815	LTF:NDS:WS:BPC 205 : 0.5997	LTF:NIS:RF:EC 205 : 0.6850	LTF:CIF:RF:UPC 4 : 0.2590	LTF:CIF:RF:UPC 3 : 0.2796	LTF:NIS:WS:BPC 20 : 0.0001	SA : LTF:CIF:RF:UPC 1 : 0.2544
4	LTF:NIS:RF:BPC 1 : 0.6731	TLF:NDS:RF:BPC 27 : 0.7174	LTF:CIF:RF:BPC 206 : 1.0383	LTF:RO:RF:BPC 218 : 1.1060	TLF:NDS:RF:BPC 25 : 0.8788	TLF:NIS:RF:BPC 2 : 0.6919	LTF:NDS:RF:BPC 43 : 0.0003	SA : LTF:NIS:RF:BPC 1 : 0.6731
5	LTF:RO:RF:BPC 1 : 0.6666	TLF:NDS:RF:BPC 29 : 0.7239	LTF:RO:WS:BPC 209 : 1.0836	LTF:NDS:WS:BPC 227 : 1.0841	TLF:NIS:WS:BPC 20 : 0.9176	TLF:NDS:RF:BPC 21 : 0.7330	LTF:NDS:RF:EC 44 : 0.0002	SA : LTF:RO:RF:BPC 1 : 0.6666
6	TLF:RO:RF:EC 5 : 0.5471	TLF:NDS:RF:EC 1 : 0.6760	LTF:RO:RF:BPC 206 : 1.2837	LTF:RO:RF:EC 216 : 1.2448	TLF:NDS:WS:BPC 17 : 0.8431	TLF:NDS:WS:BPC 13 : 0.7058	LTF:NDS:WS:BPC 6 : 0.0003	GA : TLF:NDS:RF:EC 1 : 0.6760

Table 5.24. The heuristic combinations for the optimization methods that result in lowest engineering costs with weighted-sum method.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	LTF:RO:RF:EC 0.44 : 0.2362	LTF:NIS:WS:EC 0.42 : 0.1903	LTF:NDS:RF:UPC 0.47 : 0.5748	LTF:NDS:RF:EC 0.48 : 0.5722	LTF:NDS:RF:UPC 0.41 : 0.2375	LTF:RO:RF:UPC 0.41 : 0.2544	TLF:NDS:RF:EC 0.49 : 0.0001	PSOk$_n$:LTF:NDS:RF:UPC 0.41 : 0.2375
2	TLF:NIS:RF:EC 0.35 : 0.1511	TLF:CIF:WS:EC 0.34 : 0.2167	LTF:NDS:RF:UPC 0.49 : 0.7081	LTF:NDS:WS:EC 0.50 : 0.6828	LTF:NDS:WS:EC 0.33 : 0.2461	LTF:CIF:WS:EC 0.33 : 0.2382	TLF:RO:WS:BPC 0.47 : 0.0001	PSOk$_i$:LTF:CIF:WS:EC 0.33 : 0.2382
3	TLF:NDS:WS:UPC 0.41 : 0.1633	LTF:NDS:WS:EC 0.38 : 0.2273	LTF:NDS:RF:EC 0.49 : 0.6557	LTF:NDS:RF:EC 0.51 : 0.6562	LTF:NDS:WS:EC 0.39 : 0.2907	LTF:NDS:WS:EC 0.39 : 0.2441	TLF:NDS:RF:EC 0.47 : 0.0001	GA:LTF:NDS:WS:EC 0.38 : 0.2273
4	TLF:RO:WS:EC 0.46 : 0.4568	LTF:NDS:WS:EC 0.39 : 0.6190	LTF:NDS:RF:EC 0.45 : 1.1073	LTF:NDS:RF:EC 0.46 : 1.0818	LTF:CIF:WS:EC 0.44 : 0.6446	LTF:NIS:WS:EC 0.44 : 0.6094	TLF:NDS:RF:EC 0.47 : 0.0002	GA:LTF:NDS:WS:EC 0.39 : 0.6190
5	LTF:NDS:WS:UPC 0.48 : 0.6853	LTF:CIF:WS:EC 0.40 : 0.5778	LTF:NIS:WS:EC 0.45 : 1.0589	LTF:NIS:RF:BPC 0.45 : 1.0739	LTF:CIF:WS:EC 0.43 : 0.7574	LTF:CIF:WS:UPC 0.43 : 0.7226	LTF:CIF:RF:EC 0.45 : 0.0003	GA:LTF:CIF:WS:EC 0.40 : 0.5778
6	LTF:RO:WS:UPC 0.50 : 0.6327	LTF:NDS:WS:EC 0.44 : 0.5352	LTF:CIF:RF:UPC 0.45 : 1.4197	LTF:CIF:WS:UPC 0.46 : 1.3720	LTF:NDS:WS:EC 0.47 : 0.6640	LTF:NDS:WS:EC 0.46 : 0.6289	TLF:CIF:WS:BPC 0.50 : 0.0002	GA:LTF:NDS:WS:EC 0.44 : 0.5352

Table 5.25. The heuristic combinations for the optimization methods that result in lowest engineering costs with ranking fitness method.

ds	**SA**	**GA**	**PSOp$_n$**	**PSOp$_i$**	**PSOk$_n$**	**PSOk$_i$**	**BH**	**Best Method**
1	LTF:CIF:RF:UPC 7 : 0.2400	LTF:NDS:RF:BPC 29 : 0.1926	LTF:NIS:RF:BPC 210 : 0.5184	LTF:CIF:RF:BPC 211 : 0.6699	LTF:CIF:RF:UPC 5 : 0.2847	LTF:NIS:RF:EC 1 : 0.2469	LTF:NDS:WS:BPC 31 : 0.0001	PSOk$_i$: LTF:NIS:RF:EC 1 : 0.2469
2	TLF:CIF:WS:UPC 15 : 0.1490	LTF:CIF:RF:UPC 39 : 0.1818	LTF:RO:WS:BPC 206 : 0.6004	LTF:RO:RF:BPC 228 : 0.6490	LTF:CIF:RF:EC 10 : 0.2277	LTF:CIF:WS:EC 1 : 0.2382	TLF:RO:RF:UPC 184 : 0.0001	PSOk$_i$: LTF:CIF:WS:EC 1 : 0.2382
3	LTF:CIF:RF:UPC 1 : 0.2544	LTF:CIF:RF:EC 6 : 0.1815	LTF:RO:WS:BPC 207 : 0.6062	LTF:NIS:RF:EC 207 : 0.6850	LTF:CIF:RF:UPC 4 : 0.2590	LTF:CIF:RF:UPC 3 : 0.2796	LTF:NIS:WS:BPC 8 : 0.0001	SA : LTF:CIF:RF:UPC 1 : 0.2544
4	LTF:NIS:RF:BPC 1 : 0.6731	TLF:NDS:RF:BPC 27 : 0.7174	LTF:CIF:RF:BPC 206 : 1.0383	LTF:RO:RF:EC 214 : 1.0621	TLF:NDS:RF:BPC 25 : 0.8788	TLF:NIS:RF:BPC 2 : 0.6919	LTF:NDS:RF:BPC 43 : 0.0003	SA : LTF:NIS:RF:BPC 1 : 0.6731
5	LTF:RO:RF:BPC 1 : 0.6666	TLF:NDS:RF:BPC 29 : 0.7239	LTF:RO:WS:BPC 209 : 1.0836	LTF:NDS:WS:BPC 227 : 1.0841	TLF:NIS:WS:BPC 20 : 0.9176	TLF:NDS:RF:BPC 21 : 0.7330	LTF:NDS:RF:EC 44 : 0.0002	SA : LTF:RO:RF:BPC 1 : 0.6666
6	TLF:RO:RF:EC 6 : 0.5471	TLF:NDS:RF:EC 2 : 0.6760	LTF:RO:RF:BPC 206 : 1.2837	LTF:RO:RF:EC 216 : 1.2448	TLF:NDS:WS:BPC 26 : 0.8431	TLF:NDS:WS:BPC 21 : 0.7058	LTF:NDS:WS:BPC 1 : 0.0003	BH : LTF:NDS:WS:BPC 1 : 0.0003

The weighted-sum method uses weights to find the cost of a placement, however, this may result in a lower number of mugs placed because the energy and the time for placement depend on the number of mugs. Such a case can be observed with the best method for WS in UPC, which is GA with non-increasing size, weighted sum, and user preference-based method. This method results in six mugs to be placed, whereas the maximum number of mugs placed by the other methods is ten. The fitness ranking method, on the other hand, allows the parameters to be ranked as they were by the participants in our experiments. Therefore, the method compares each parameter in the order of their ranks and as a result, it would find the best solution that would satisfy all the criterion. Hence, we decided to use ranking fitness method with user preference-based function, such that all our parameters including the sortedness parameter can be considered within the ranks of their importance. As a result, for the general case, we can conclude that the Simulated Annealing with Left-Top-Fill heuristic with non-increasing sort method as the initial configuration and the ranking fitness cost method with the user preference-based function is the best approach for our dishwasher problem, as it occurs to be the best solution in different cost functions in the datasets.

The results of the best placement method and heuristic combinations for the user-preference based cost within the ranking fitness method are presented in Figure 5.9 and Figure 5.10. As can be observed from Figure 5.10 in the sixth dataset, GA was the best solution that found a solution with a sorted order, as a result of using the ranking fitness method with the user-preference based cost.

5.3.3. Robot Experiments

We conducted preliminary tolerance tests as described in Section 5.3.1, and we compared the initial configuration methods to SA optimization using the real robot, in which we observed that we had to use a force-sensing resistor in order to understand the grasp of a mug. The video of the experiments is available online [111].

We use the ranking fitness method with the user-preference based cost for the experiments with our robots, for the reasons that are described in the previous section. We use

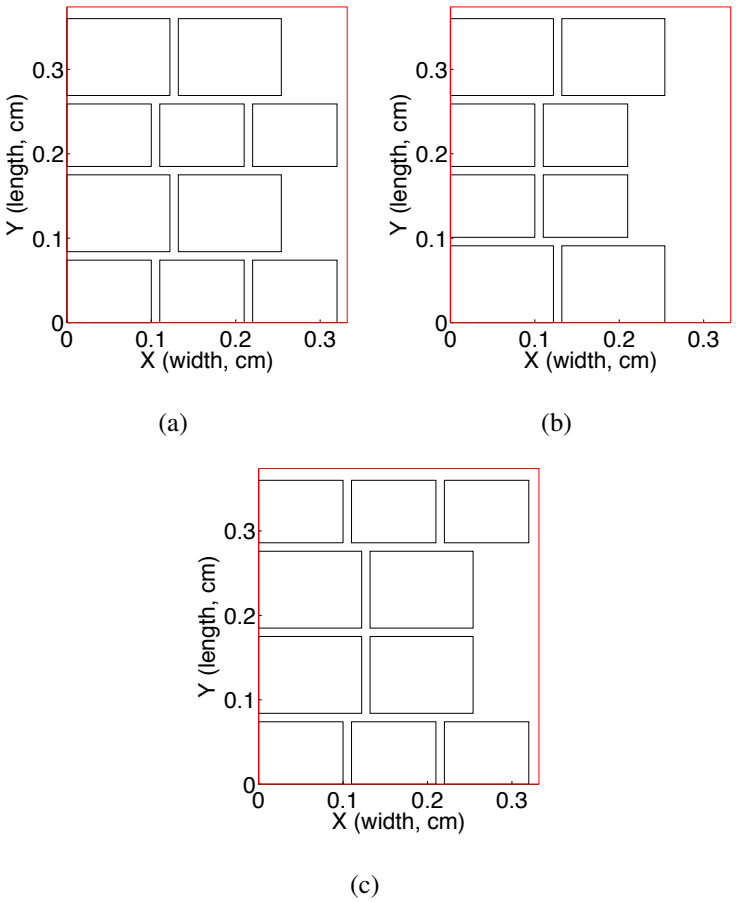
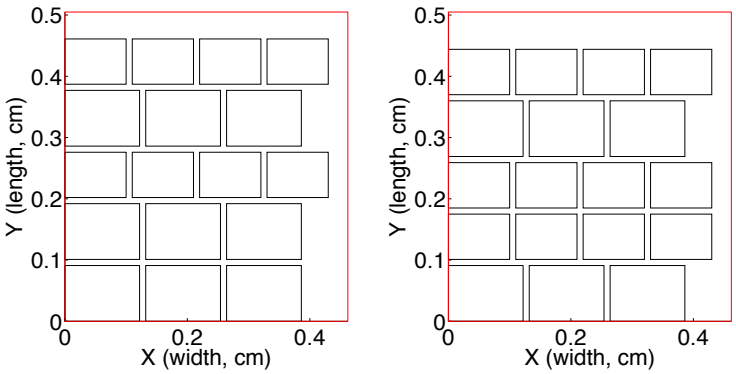
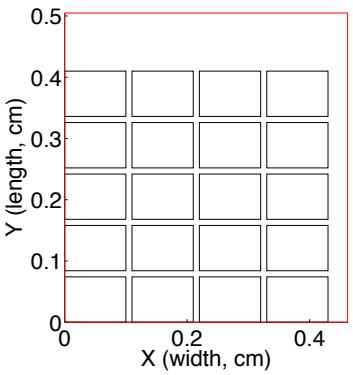


Figure 5.9. The placements that correspond to the lowest user-preference based cost within the ranking fitness method : (a) dataset 1 with GA:TLF:CIF:WS:BPC (ten placed mugs), (b) dataset 2 with PSOk$_i$:LTF:NIS:RF:UPC (eight placed mugs), (c) dataset 3 with SA:LTF:CIF:RF:UPC (ten placed mugs).



(a)

(b)



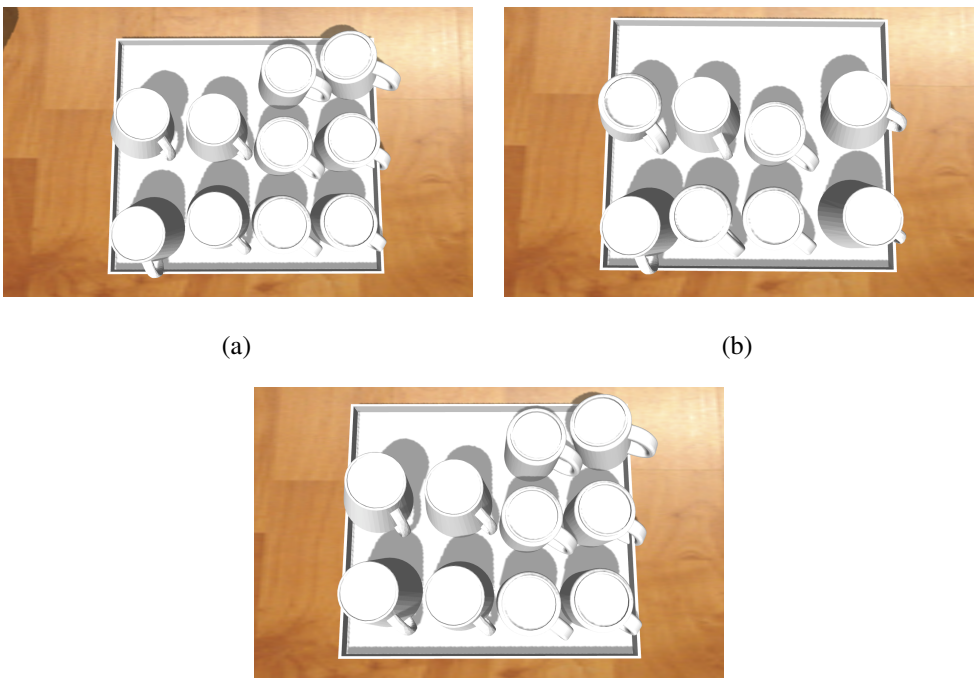
(c)

Figure 5.10. The placements that correspond to the lowest user-preference based cost within the ranking fitness method : (a) dataset 4 with SA:LTF:NIS:RF:BPC (17 placed mugs), (b) dataset 5 with SA:LTF:NIS:RF:BPC (18 placed mugs), (c) dataset 6 with GA:TLF:NDS:RF:EC (20 placed mugs).

Gazebo for the experiments, but object recognition results are obtained through the depth camera. The best method combinations for each dataset obtained in the previous section were used in the robot experiments. The results of the experiments in *Gazebo* are presented in Table 5.26. The placement images for each dataset in which all of the mugs are detected, are given in Figure 5.11. The video of the placement experiments in *Gazebo* is available online [112].

Table 5.26. The results of the experiments within Gazebo.

ds	**Number of Mugs Detected**	**Number of Mugs Placed**	**Placement Time (min.)**
1	10.50±1.92	9.00±0.22	9.13±1.12
2	10.50±1.92	7.50±0.5	7.68±0.50
3	10.75±1.20	9.10±1.00	9.44±0.95



(a) (b)

(c)

Figure 5.11. The placements for the experiments in Gazebo: (a) dataset 1 with ten placed mugs, (b) dataset 2 with eight placed mugs, (c) dataset 3 with ten placed mugs.

In the Gazebo experiments, we observed that the mug handles were oriented at differ-

ent angles for different positions since the robot arm does not have a wrist yaw joint. This resulted in the mugs to shift in position in negative Y direction. Therefore, the mugs at the left-most and the right-most positions could not fit properly into the tray. Also the error in kinematics, which is a function of x, y positions as described in Appendix A, affected the positions of the mugs within the tray. Moreover, we observed that the Top-Left or Left-Top placement orders of the mugs might not be maintained because of the ordering of the mugs according to the coordinates and not the relative ordering within the tray. For example, a mug with a higher x but a lower y coordinate than another mug will be placed prior to the other mug, but this might cause a collision with the gripper when the other mug is being placed to the left, as shown in Figure 5.12. The relative ordering of the mugs within the tray according to the Top-Left or Left-Top placements should be used in order to prevent any collisions.

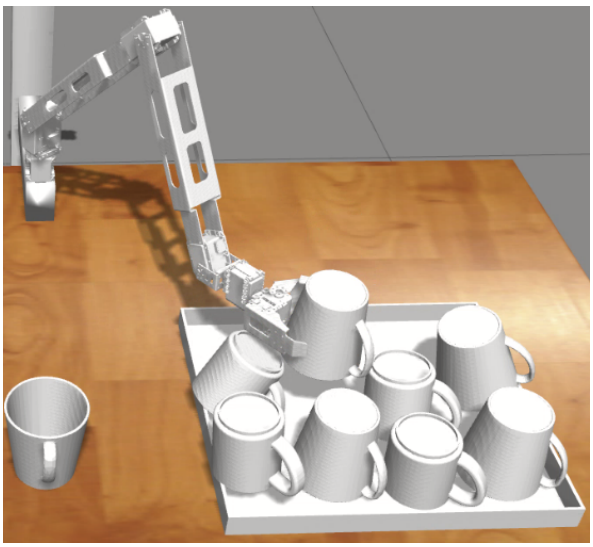


Figure 5.12. Collision of the gripper with the previously placed mug.

If we compare our results with the observational study, we can see that a lower number of mugs is placed for each dataset with our optimization methods. There are three reasons for this: we use a tolerance between the mugs in order to achieve collision-free placements, as we derived from the tolerance tests for our robot. Furthermore, the wrist yaw joint is not available in our robot arm, which prevents the arm to rotate objects in the side grasp. Finally, we use bounding boxes as rectangles for our mugs, which prevents placing the mug handles at different orientations, such that mugs are packed as close as possible. If a different robot arm with at least 6 DOF is used, the first two problems might be eliminated. In that case, a

tessellation method can be used instead of rectangular bounding boxes, since rotations will be possible. The robot takes a significantly longer amount of time than the average placement time for the participants. However, it can place ten mugs within nine-ten minutes, therefore, when compared to the questionnaire used in the annotation, it would be accepted by 29.1% of the annotators. In our approach, we use sorting methods to find an initial configuration of the objects. In other words, our approach is similar to the approach of the people in the observation study, in that, a sorted-size ordering is preferred if all the mugs are placed.

We could not conduct the final experiments on the real robot due to problems with the motors. The gripper motor fails with the overload error when the force-sensing resistor is used to understand the grasp. Furthermore, the gripper motor fails occasionally with the same error without the FSR, therefore, the experiment could not be completed. The reason for the error might be due to the limit effort of the gripper motor, such that the motor could not lift a plastic mug. There might also be a wear problem with the motor since it worked normally in the experiments in Section 5.3.1.

5.4. Comparison of Our Experimental Setting to a Real Kitchen

We test our robot in an environment in which there are only mugs and a tray. There were no other obstacles which are normally available in a kitchen setting, such as, other dishes, sink, jars, or any kitchen utensil. In order to add such a feature, object recognition algorithm should be improved in order to add these objects into consideration. Furthermore, manipulation planning should be aware of these obstacles, which can be adjusted in our algorithm by changing the kinematics calculation module in *ROS MoveIt!*, which accounts for the obstacles detected automatically. However, this would prolong the time required for manipulation, due to the kinematics calculation time.

In our study, we were not concerned with the safety factor, that is, it is not “safe” for other humans to be around our robot arm since we did not detect humans in our object detection algorithm. In other words, if a person puts his or her hand in front of the robot during manipulation, the arm could hit him or her. The object detection algorithm should be improved in order to detect humans, and the arm should either avoid them or stop until the

human is out of the working space of the robot.

The tray we are using has a flat surface, whereas a normal dishwasher tray has an uneven surface along with holders. Therefore, these should be detected and taken into consideration in the placement algorithm. However, if the number of "columns" in the tray for the uneven surfaces, and the number of "cells" for the holders can be obtained from the object detection, the number of columns and rows within the occupancy matrix in the placement planning can be adjusted for those given parameters, without changing any method in our algorithm. In that case, the number of mugs placed may decrease since it would not be possible to place mugs in between two uneven surfaces at different levels. Manipulation planning would not be affected because the gripper releases the mugs from above the tray, hence, it would not collide with the holders.

Due to the restrictions of a relatively small configuration space and a fixed base, our tray was on the counter-top next to the mugs. In a real life setting, the tray would be below the counter-top or at a further location than the counter-top. Which was why we initially thought to use the arm with a mobile base and modeled with it in *Gazebo* simulation environment. However, when the arm was attached to the mobile base, the motion was unstable, i.e. the arm vibrated during manipulation, which was an undesirable situation that could cause the mugs to drop, hence we detached the arm from its base to be fixed on the tabletop, and modified the *Gazebo* model in order to adjust for the changes in the system. If a stable mobile system using the arm and the camera can be made, then either an image stitching algorithm should be used in the object detection or the positions of the detected mugs and the tray can be recorded on a map of the environment, and the motion of the robot arm can be processed on that map. Using a mobile base would increase the number of mugs placed, hence, the robot would be able to place as many mugs as in our simulated results.

The lack of a wrist yaw joint in our arm caused the mugs to be slightly rotated in the tray, that is, their mug handles were not exactly parallel to the X direction. We also could not rotate the mugs from the top, because our configuration space decreased in the Y direction as x increases for the top grasp. Therefore, we omitted rotation of the mugs in order to be able to use a bigger size tray. However, because we modeled our mugs with rectangles

which allowed empty spaces around the mug, the undesired orientation of the mugs did not result in a collision between mugs. If a 6-DOF robot is used, then the mugs can be given any rotation. In that case, the placement algorithm should model mugs as hexagonal shapes, such that a better tessellation can be made. Also, the orientation should be a parameter for the optimization methods.

Finally, we assumed that the mugs on the counter-top were in an upright position, which is a valid assumption in a real world setting, in general. However, the upside-down mugs can already be detected as they are on the tray, so they can be detected on the counter-top as well. In that case, the upside-down mugs can be placed directly onto the tray or they do not have to be turned upside-down again over the empty area. The mugs which are fallen to their side can also be detected if the object detection algorithm is trained, and during our experiments, we observed that the fallen mugs can be turned upright with our arm.

6. CONCLUSION

In this thesis, we developed a fully autonomous robot system, that consists of a 3D depth camera and a 5 Degree-of-Freedom robot arm, which can place the mugs on the counter-top onto a dishwasher tray. Our approach combined object recognition, placement planning, and manipulation planning. A placement planning using different initial placement methods with sorted sizes or order is made with Simulated Annealing, Genetic Algorithm, and Particle Swarm Optimization methods in order to find a near-optimal placement within the tray. We use a force-sensing resistor, in order to perceive successful grasps, and also to understand when the mug is dropped during manipulation. We use an empty area on the counter-top in order to turn our mugs upside down in the manipulation planning, because the other approach, which was placing the closest item first direct from the position on the counter-top to the tray, would either cause collisions with previously placed mugs or lead to a smaller sized tray in the case of top grasps for placing.

We conducted preliminary and user studies in order to find a cost function for our placement. After the preliminary study, we found out that our approach satisfies some of the criteria, which included not stacking and not breaking the mugs and placing them upside-down on the upper tray, by our assumptions and grasp planning. We observed that we could optimize four of the criteria from the results of the study, which were using the maximum capacity of the dishwasher tray by the number of mugs placed and the covered area, along with the energy spent during placement, placement time and packing similar sized objects together. Hence, we used these parameters in our user study in order to find their weights, and we concluded that the number of mugs placed was the most important criteria, followed by energy use, time and sortedness of the placement. We further conducted an observational study in order to observe the placement methods of the people when they are placing their dishes, and to find the optimum placements for our datasets. Prior to the experiments in the study, we asked our participants to fill the questionnaire we used in the user study. In this study, we asked our participants to place the mugs on the counter-top to the given tray. We used six datasets with different number of mugs and mug type ratios. We recorded the placement times and the number of mugs placed for each dataset. We observed that

the participants used preferred technique of placing items, if they can fit all the mugs with that method. If they cannot, then they either change their method or rearrange mugs. In the datasets that has more small mugs, NDS heuristic is preferred, whereas, for the other datasets, most of the participants preferred NIS. We apply the same approach as our participants in our algorithms, that is, if all the mugs can be placed with an initial sorted configuration, then further optimization is not made.

After the experiments, the participants filled an Obsessive-Compulsive Scale test, such that we can find a relation between OCD and the sortedness parameter if any exists. The placements of the participants were annotated with a 5-point Likert scale in the annotation study. In this study, we also asked our annotators to fill the questionnaires in the observational study. The weights for the cost functions were derived from the results of the annotation questionnaire, in which the number of mugs placed was the most important criterion, followed by energy, time, and sortedness as in the user study. The results of the annotation showed that more than 50% of the participants who ranked sortedness as important placed in a moderately-sorted way, and the percentage of participants that make a sorted placement decreases when the solution to the packing problem cannot be solved optimally by a sorted placement. Therefore, we concluded that the sortedness parameter was not very important in the placement, hence, it can be ignored for the weighted-sum method, such that a placement with a higher number of mugs can be found that is not in a sorted order. However, the fitness ranking method compares the sortedness parameter as the last comparison, therefore, it provides a better solution with a lower cost than the best weighted-sum method. We compared the OCD states of the annotators and the participants, and we found that OCD affects both the rank and the sorted placement of the people. However, OCD is not the only reason behind the sortedness parameter, as people prefer to place items in a sorted order, for the ease of unloading and also because they think that a sorted order will result in maximizing the capacity.

We compared our optimization methods with different heuristics and cost functions. We observed that, in general, SA with LTF packing heuristic provides the best results for the ranking fitness functions and GA with LTF packing heuristic provides the best results for the weighted-sum cost functions. The improved PSOk algorithm provides lower costs than the

normal PSOk. The highest costs with the lowest number of placed mugs were found through the PSOp methods. At certain datasets, BH heuristics provide the lowest cost, however, it was unable to find a near-optimal solution in the second dataset. Comparing the best methods for the user-preference functions in the weighted-sum and the ranking fitness methods, we found that the best optimization method in the weighted-sum method results in lower number of mugs placed, in order to decrease the energy and time required by the function. Therefore, we concluded that the ranking fitness method with the user preference-based cost gives the best result for our problem, as all the parameters are compared in a ranked order. Finally, we concluded that Simulated Annealing with the Left-Top-Fill heuristic with non-increasing size sort method as the initial configuration and the ranking fitness method with the user preference-based cost function is the best approach for our dishwasher problem. We used the best methods for each our datasets from the simulations and applied them to the *Gazebo* simulation environment, in which our robot is fully configured. The results varied depending on the number of objects detected at an iteration. We observed that the approximated placement time in the simulation represented the actual placement time well. Furthermore, we saw that we had to consider the relative placement ordering of the mugs within the tray, instead of the mug positions, in order to avoid collisions with the gripper. Due to the gripper motor failure during the experiments, we could not test our approach on the real robot.

6.1. Future Work

This project is expandable in terms of several aspects in order to make our system be more suitable for use at home. First of all, we plan on testing our approach on another robot, since the motors of our robots had problems, which prevented us from conducting the experiments. Furthermore, we plan on adding an interface to the system, such that a person can start the robot through the interface. This interface will have two options: general settings that use the weights derived from the annotation study for the cost function, and custom settings that asks the user to enter their preferred rank of importance of the parameters, as we asked in our user studies. The custom settings will allow for the ranking order to be changed in the ranking fitness method for the user preference-based cost function, such that a solution can be found in the optimization method that would be suitable for the preferences of

the person. In this study, we found that Simulated Annealing with the Left-Top-Fill heuristic is the overall most frequently selected best method in all of our cost functions for the ranking fitness method, therefore, we do not need to change the optimization algorithm. We believe that the robots that are used at houses should be adjustable according to the preferences of people such that they would be useful in terms of fulfilling the demands of their user.

As we noted in Section 5.4, different types of dishes should be detected in a real kitchen, to place them into the dishwasher. Therefore, manipulation planning needs to be adjusted in order to grasp different types of dishes, such as plates, which cannot be grasped from the top. Furthermore, object recognition algorithm needs to be improved to detect these dishes and the dynamic and stationary obstacles in a kitchen environment. An obstacle avoidance method can be implemented or the available modules on *ROS MoveIt!* can be used for collision-free path planning for manipulation. Moreover, a person should be detected for safety reasons. In order to place a dish into an actual dishwasher tray, the racks and the holders need to be detected for limiting the available positions for the placement of an item in placement planning.

Our approach can be implemented on another robot that has at least 6 degrees-of-freedom, such that the items can be rotated within the dishwasher tray or prior to that. This future directive brings about the need to adjust our placement planning approach. The orientation of an item should be considered along with the packing order of the item for creating a neighboring solution. In that case, a probability can be used to change either the orientation of the item, or its packing order, in all of our optimization methods, as adopted in the literature for the bin-packing problems with rotatable items. Furthermore, the items can be represented with shapes other than rectangles, in order to reduce the lost area between the mugs. Tessellation methods can be used for the packing heuristics.

Finally, our system solution can be used with a mobile base to enable mobile manipulation, similar to the related work presented in Chapter 1. An image stitching algorithm can be used in object recognition or mapping of the objects and the robot can be used to track their positions.

REFERENCES

1. Maytronics, "Dolphin Robotic Pool Cleaners", `http://www.maytronics.com/robotic-pool-cleaners`, accessed at March 2016.

2. iRobot, "Roomba", `http://www.irobot.com/us/learn/home/roomba.aspx`, accessed at March 2016.

3. Kopicki, M., R. Detry, F. Schmidt, C. Borst, R. Stolkin and J. L. Wyatt, "Learning Dexterous Grasps That Generalise to Novel Objects by Combining Hand and Contact Models", *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5358–5365, IEEE, 2014.

4. Saxena, A., J. Driemeyer and A. Y. Ng, "Robotic Grasping of Novel Objects Using Vision", *The International Journal of Robotics Research (IJRR)*, Vol. 27, pp. 157–173, 2008.

5. Dillmann, R., "Teaching and Learning of Robot Tasks via Observation of Human Performance", *Robotics and Autonomous Systems*, Vol. 47, pp. 109–116, 2004.

6. Hsiao, K., P. Nangeroni, M. Huber, A. Saxena and A. Y. Ng, "Reactive Grasping Using Optical Proximity Sensors", *2009 IEEE International Conference on Robotics and Automation*, pp. 2098–2105, IEEE, Kobe, Japan, 2009.

7. Okada, K., M. Kojima, Y. Sagawa, T. Ichino, K. Sato and M. Inaba, "Vision Based Behavior Verification System of Humanoid Robot for Daily Environment Tasks", *6th IEEE-RAS International Conference on Humanoid Robots*, pp. 7–12, 2006.

8. The Telegraph, "Kitchen Robot Loads the Dishwasher", `http://www.telegraph.co.uk/technology/3891631/Kitchen-robot-loads-the-dishwasher.html`, accessed at March 2016.

9. Inaba, M., K. Okada, T. Yoshikai, R. Hanai and K. Yamazaki, "Enhanced Mother Environment with Humanoid Specialization in IRT Robot Systems", *The 14th International Symposium on Robotics Research (ISRR)*, Vol. 70 of *Springer Tracts in Advanced Robotics*, pp. 379–394, Springer, Lucerne, Switzerland, 2011.

10. Vahrenkamp, N., T. Asfour and R. Dillmann, "Simultaneous Grasp and Motion Planning", *IEEE Robotics and Automation Magazine*, Vol. 19, No. 2, pp. 43–57, 2012.

11. Jiang, Y., C. Zheng, M. Lim and A. Saxena, "Learning to Place New Objects in a Scene", *International Journal of Robotics Research (IJRR)*, Vol. 31, pp. 1021–1043, 2012.

12. BBC News, "Boris the Robot Can Load Up Dishwasher", `http://www.bbc.com/news/science-environment-29168675`, accessed at March 2016.

13. Khil, A.-R. and K.-H. Lee, "Optimization of a Robot-Served Cart Capacity Using the Three-Dimensional Single Bin Packing Problem", *Multimedia Tools and Applications*, Vol. 74, No. 1, pp. 185–198, 2014.

14. Lee, K.-H. and A.-R. Khil, "Optimization of a Dish Cart Storage using the Two-Dimensional Single Bin Packing Problem", *Applied Mathematics & Information Sciences*, Vol. 8, No. 5, pp. 2229–2236, 2014.

15. Lodi, A., S. Martello and M. Monaci, "Two-Dimensional Packing Problems: a Survey", *European Journal of Operational Research*, Vol. 141, pp. 241–252, 2002.

16. Mhaiskar, N. D., "Two-Dimensional Rectangle Packing Problems: A Survey", *International Journal of Computer, Information Technology & Bioinformatics (IJCITB)*, Vol. 2, No. 1, pp. 10–15, 2014.

17. Baker, B. S., E. G. J. Coffman and R. L. Rivest, "Orthogonal Packing in Two Dimensions", *SIAM Journal on Computing*, Vol. 9, pp. 846—-855, 1980.

18. Chazelle, B., "The Bottom-Left Bin-Packing Heuristic: an Efficient Implementation", *IEEE Transactions on Computers*, Vol. C-32, No. 8, pp. 697–707, 1983.

19. Hopper, E. and B. C. H. Turton, "A Genetic Algorithm for a 2D Industrial Packing Problem", *Computers and Industrial Engineering*, Vol. 37, No. 1, pp. 375–378, 1999.

20. Hopper, E. and B. C. H. Turton, "Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D Packing Problem", *European Journal of Operational Research*, Vol. 128, No. 1, pp. 34–57, 2001.

21. Burke, E. K., G. Kendall and G. Whitwell, "A New Placement Heuristic for the Orthogonal Stock-Cutting Problem", *Operations Research*, Vol. 52, No. 4, pp. 655–671, 2004.

22. Imahori, S., M. Yagiura and H. Nagamochi, *Practical Algorithms for Two-Dimensional Packing*, pp. 36:1–36:14, Computer & Information Science Series, Chapman & Hall/CRC, 2006.

23. Riff, M. C., X. Bonnaire and B. Neveu, "A Revision of Recent Approaches for Two-Dimensional Strip-Packing Problems", *Engineering Applications of Artificial Intelligence*, Vol. 22, pp. 833–837, 2009.

24. Hochbaum, D. S. and W. Maass, "Approximation Schemes for Covering and Packing problems in Image Processing and VLSI", *Journal of the ACM*, Vol. 32, No. 1, pp. 130–136, 1985.

25. Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, No. 4598, pp. 671–680, 1982.

26. Dowsland, K. A., "Some Experiments with Simulated Annealing Techniques for Packing Problems", *European Journal of Operational Research*, Vol. 68, No. 3, pp. 389–399, 1993.

27. Dereli, T. and G. Sena Daş, "A Hybrid Simulated-Annealing Algorithm for Two-Dimensional Strip Packing Problem", *Adaptive and Natural Computing Algorithms: 8th International Conference, ICANNGA*, pp. 508–516, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

28. Zhang, D., Y. Kang and A. Deng, "A New Heuristic Recursive Algorithm for the Strip Rectangular Packing Problem", *Computers & Operations Research*, Vol. 33, No. 8, pp. 2209–2217, 2006.

29. Egeblad, J. and D. Pisinger, "Heuristic Approaches for the Two and Three-Dimensional Knapsack Packing Problem", *Computers & Operations Research*, Vol. 36, No. 4, pp. 1026–1049, 2009.

30. Peng, Y., D. Zhang and F. Y. L. Chin, "A Hybrid Simulated Annealing Algorithm for Container Loading Problem", *The first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09)*, pp. 919–922, ACM, Shanghai, China, 2009.

31. Bischoff, E. E. and M. S. W. Janetz, F.and Ratcliff, "Loading Pallets with Non-Identical Items", *European Journal of Operational Research*, Vol. 84, No. 3, pp. 681–692, 1995.

32. Bischoff, E. E. and M. S. W. Janetz, F.and Ratcliff, "Issues in the Development of Approaches to Container Loading", *Omega, International Journal of Management Science*, Vol. 23, No. 4, pp. 377–390, 1995.

33. Gehring, H. and A. Bortfeldt, "A Genetic Algorithm for Solving the Container Loading Problem", *International Transactions in Operational Research*, Vol. 4, No. 5-6, pp. 401–418, 1997.

34. Bortfeldt, A. and H. Gehring, "A Hybrid Genetic Algorithm for the Container Loading Problem", *European Journal of Operational Research*, Vol. 131, No. 1, pp. 143 – 161, 2001.

35. Bortfeldt, A., H. Gehring and D. Mack, "A Parallel Tabu Search Algorithm for Solving

the Container Loading Problem", *Parallel Computing*, Vol. 29, No. 5, pp. 641–662, 2003.

36. Dereli, T. and G. Sena Daş, "A Hybrid Simulated Annealing Algorithm for Solving Multi-objective Container-Loading Problems", *Applied Artificial Intelligence*, Vol. 24, No. 5, pp. 463–486, 2010.

37. George, J. A. and D. F. Robinson, "A Heuristic for Packing Boxes into a Container", *Computers & Operations Research*, Vol. 7, pp. 147—156, 1980.

38. Wang, H., Z. Wang and J. Luo, "A Simulated Annealing Algorithm for Single Container Loading Problem", *9th International Conference on Service Systems and Service Management (ICSSSM)*, pp. 551–556, 2012.

39. Can, O. and O. K. Şahingöz, "Solving Container Loading Problem with Simulated Annealing Algorithm", *IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 379–383, 2014.

40. Holland, J., *Adaptation in Natural and Artificial Systems*, Complex Adaptive Systems, University of Michigan Press, 1975.

41. Kröger, B., "Guillotineable Bin Packing: a Genetic Approach", *European Journal of Operational Research*, Vol. 84, No. 3, pp. 645–661, 1995.

42. Jakobs, S., "On Genetic Algorithms for the Packing of Polygons", *European Journal of Operational Research*, Vol. 88, No. 1, pp. 165–181, 1996.

43. Gehring, H. and A. Bortfeldt, "A Parallel Genetic Algorithm for Solving the Container Loading Problem", *International Transactions in Operational Research*, Vol. 9, No. 4, pp. 497–511, 2002.

44. Bortfeldt, A., "A Genetic Algorithm for the Two-Dimensional Strip Packing Problem with Rectangular Pieces", *European Journal of Operational Research*, Vol. 172, No. 3,

pp. 814–837, 2006.

45. Wu, Y., W. Li, M. Goh and R. de Souza, "Three-Dimensional Bin Packing Problem with Variable Bin Height", *European Journal of Operational Research*, Vol. 202, No. 2, pp. 347–355, 2010.

46. Pintea, C., C. Pascan and M. Hajdu-Macelaru, "Comparing Several Heuristics for a Packing Problem", *Int J. Advanced Intelligence Paradigms*, Vol. 4, No. 3, pp. 268–277, 2012.

47. Kennedy, J. and R. Eberhart, "Particle Swarm Optimization", *Proceedings of the 1995 IEEE the International Conference on Neural Networks*, Vol. 6, pp. 1942—1948, 1995.

48. Zhao, C., L. Lin, C. Hao and L. Xinbao, "Solving the Rectangular Packing Problem of the Discrete Particle Swarm Algorithm", *Proceedings of the 2008 International Seminar on Business and Information Management*, Vol. 2 of *ISBIM '08*, pp. 26–29, IEEE Computer Society, Washington, DC, USA, 2008.

49. He, C., Y.-B. Zhang, J.-W. Wu and C. Chang, "Research of Three-Dimensional Container-Packing Problems Based on Discrete Particle Swarm Optimization Algorithm", *International Conference on Test and Measurement (ICTM) '09*, Vol. 2, pp. 425–428, 2009.

50. Yang, Q. and W. Jin Min, "The Particle Swarm Optimization Algorithm for Solving Rectangular Packing Problem", *Advanced Materials Research*, Vol. 186, pp. 479–483, 2011.

51. Shin, Y.-B. and E. Kita, "Solving Two-Dimensional Packing Problem Using Particle Swarm Optimization", *Computer Assisted Methods in Engineering and Science*, Vol. 19, pp. 241—255, 2012.

52. Shin, Y.-B. and E. Kita, "Search Performance Improvement of Particle Swarm Optimization by Second Best Particle Information", *Applied Mathematics and Computa-*

tion, Vol. 246, pp. 346–354, 2014.

53. Domingo, B., S. Ponnambalam and G. Kanagaraj, "Particle Swarm Optimization for the Single Container Loading Problem", *IEEE International Conference on Computational Intelligence Computing Research (ICCIC)*, pp. 1–6, 2012.

54. Chen, W., P. Zhai, H. Zhu and Y. Zhang, "Hybrid Algorithm for the 2-D Rectangular Layer Packing Problem", *Journal of the Operational Research Society*, Vol. 65, No. 7, pp. 1068–1077, 2014.

55. Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence", *Comput. Oper. Res.*, Vol. 13, No. 5, pp. 533–549, 1986.

56. Gendreau, M., M. Iori, G. Laporte and S. Martello, "A Tabu Search Algorithm for a Routing and Container Loading Problem", *Transportation Science*, Vol. 40, No. 3, pp. 342–350, 2006.

57. Lodi, A., S. Martello and D. Vigo, "Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems", *INFORMS Journal on Computing*, Vol. 11, No. 4, pp. 345–357, 1999.

58. Viegas, J. L., S. M. Vieira, E. M. P. Henriques and J. M. C. Sousa, "A Tabu Search Algorithm for the 3D Bin Packing Problem in the Steel Industry", *Proceedings of the 11th Portuguese Conference on Automatic Control (CONTROLO'2014)*, pp. 355–364, Springer International Publishing, 2015.

59. Feo, T. A. and M. G. C. Resende, "A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem", *Oper. Res. Lett.*, Vol. 8, No. 2, pp. 67–71, 1989.

60. Moura, A. and J. F. Oliveira, "A GRASP Approach to the Container-Loading Problem", *IEEE Intelligent Systems*, Vol. 20, No. 4, pp. 50–57, 2005.

61. Parreño, F., R. Alvarez-Valdes, J. M. Tamarit and J. F. Oliveira, "A Maximal-Space

Algorithm for the Container Loading Problem", *Institute for Operations Research and the Management Sciences J. on Computing*, Vol. 20, No. 3, pp. 412–422, 2008.

62. Alvarez, M., R. Alvarez-Valdes and F. Parreño, "A GRASP Algorithm for the Container Loading Problem With Multi-Drop Constraints", *Pesquisa Operacional*, Vol. 35, pp. 1–24, 2015.

63. Ren, X., D. Fox and K. Konolige, "Change Their Perception: RGB-D for 3-D Modeling and Recognition", *IEEE Robotics Automation Magazine*, Vol. 20, No. 4, pp. 49–59, 2013.

64. Maitin-Shepard, J., M. Cusumano-Towner, J. Lei and P. Abbeel, "Cloth Grasp Point Detection Based on Multiple-view Geometric Cues with Application to Robotic Towel Folding", *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2308–2315, 2010.

65. Doumanoglou, A., A. Kargakos, T.-K. Kim and S. Malassiotis, "Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning", *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 987—-993, 2014.

66. Lenz, I., R. A. Knepper and A. Saxena, "DeepMPC: Learning Deep Latent Features for Model Predictive Control", *Robotics: Science and Systems (RSS)*, pp. 987–993, 2015.

67. Collet-Romea, A., M. Martinez-Torres, S. Srinivasa and D. Ferguson, "The MOPED Framework: Object Recognition and Pose Estimation for Manipulation", *International Journal of Robotics Research*, Vol. 30, No. 10, pp. 1284–1306, 2011.

68. Browatzki, B., V. Tikhanoff, G. Metta, H. H. Bülthoff and C. Wallraven, "Active Object Recognition on a Humanoid Robot", *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2021–2028, 2012.

69. Kehoe, B., A. Matsukawa, S. Candido, J. Kuffner and K. Goldberg, "Cloud-based

Robot Grasping with the Google Object Recognition Engine", *2013 IEEE Int'l Conf. on Robotics and Automation*, pp. 1–8, 2013.

70. Kent, D., M. Behrooz and S. Chernova, "Construction of a 3D Object Recognition and Manipulation Database from Grasp Demonstrations", *Autonomous Robots*, Vol. 40, No. 1, pp. 175–192, 2016.

71. Kornuta, T. and M. Laszkowski, "Perception Subsystem for Object Recognition and Pose Estimation in RGB-D Images", *Challenges in Automation, Robotics and Measurement Techniques: Proceedings of AUTOMATION-2016*, pp. 597–607, Springer International Publishing, 2016.

72. Guo, Y., M. Bennamoun, F. Sohel, M. Lu and J. Wan, "3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 11, pp. 2270–2287, 2014.

73. Open Source Robotics Foundation, "Robot Operating System", `http://www.ros.org/`, accessed at March 2016.

74. Willow Garage, ROS community, "ORK: Object Recognition Kitchen", `https://github.com/wg-perception/object_recognition_core`, accessed at March 2016.

75. Waibel, M., M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle and R. van de Molengraft, "RoboEarth", *Robotics Automation Magazine*, Vol. 18, No. 2, pp. 69–82, 2011.

76. Itseez, "OpenCV", `http://opencv.org`, accessed at March 2016.

77. Rusu, R. B. and S. Cousins, "3D is Here: Point Cloud Library (PCL)", *2011 IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

78. Ozgur, A. and H. L. Akin, "Planning for Tabletop Clutter Using Affordable 5-DOF Manipulator", *2nd Workshop on Robots in Clutter: Preparing Robots for the Real World*, Berlin, Germany, 2013.

79. Open Source Robotics Foundation, "Gazebo", `http://gazebosim.org`, accessed at March 2016.

80. Sucan, I. A. and S. Chitta, "MoveIt!", `http://moveit.ros.org`, accessed at March 2016.

81. ASUS, "ASUS Xtion Pro Live", `http://www.asus.com/Multimedia/Xtion_PRO_LIVE`, accessed at March 2016.

82. Robotis, "Robotis", `http://www.robotis.com/xe/dynamixel_en`, accessed at March 2016.

83. Robotis, "USB2Dynamixel", `http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm`, accessed at March 2016.

84. Denavit, J. and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *Trans. of the ASME Journal of Applied Mechanics*, Vol. 23, pp. 215–221, 1955.

85. Spong, M. W., S. Hutchinson and M. Vidyasagar, *Forward Kinematics: The Denavit-Hartenberg Convention*, pp. 3:61–3:81, John Wiley & Sons, Inc., 2005.

86. Yıldırım, Y., O. Aşık, B. Görer, N. E. Özkucur and H. L. Akın, "Tur Rehberi Çoklu Robot Sistemi", *Türkiye Otonom Robotlar Konferansı*, Ankara, Turkey, 2014, in Turkish.

87. Festo, "Robotino", `http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino`, accessed at March 2016.

88. Hokuyo Automatic Co. Ltd, "URG-04LX Laser Scanner", `https://www.hokuyo-`

aut.jp/02sensor/07scanner/urg_04lx.html, accessed at March 2016.

89. Robotistan, "Pololu 1.5 Inch Kuvvete Duyarlı Kare Sensör - Force-Sensing Resistor", http://www.robotistan.com/15-kuvvete-duyarli-kare-sensor-force-sensing-resistor-15-square, accessed at March 2016.

90. Shukla, D., O. Erkent and J. Piater, "Probabilistic Detection of Pointing Directions for Human Robot Interaction", *International Conference on Digital Image Computing: Techniques and Applications*, pp. 1–8, IEEE, Adelaide, SA, 2015.

91. Sucan, I. A., "URDF", http://wiki.ros.org/urdf, accessed at March 2016.

92. Inc, D. S., *Stereolithography Interface Specification*, 1988.

93. Open Source Solid Modeling, "BRL-CAD", http://brlcad.org/, accessed at March 2016.

94. Orocos, "Orocos Kinematics and Dynamics", http://www.orocos.org/kdl, accessed at March 2016.

95. MathWorks, "MATLAB", http://www.mathworks.com/products/matlab/, accessed at March 2016.

96. Hershberger, D., D. Gossow and J. Faust, "RViz", http://wiki.ros.org/rviz, accessed at March 2016.

97. Sucan, I. A., "SRDF", http://wiki.ros.org/srdf, accessed at March 2016.

98. Evans, C. C., "YAML: YAML Ain't Markup Language", http://yaml.org/, accessed at March 2016.

99. Rebguns, A., "Dynamixel_motor", https://github.com/arebgun/dynamixel_motor, accessed at March 2016.

100. Arduino, “Arduino Board Uno”, https://www.arduino.cc/en/Main/ArduinoBoardUno, accessed at March 2016.

101. ROS, “Intrinsic Calibration of the Kinect Cameras”, http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration, accessed at March 2016.

102. Dassault Systemes, “3D CAD Design Software SOLIDWORKS”, http://www.solidworks.com/, accessed at March 2016.

103. ROS, “Recording and Playing Back Kinect Data”, http://wiki.ros.org/openni_launch/Tutorials/BagRecordingPlayback, accessed at March 2016.

104. Bogazici University, “İnsan Araştırmaları Kurumsal Değerlendirme Kurulu”, http://www.boun.edu.tr/tr-TR/Content/Genel/Yonetim/Kurul_ve_Komisyonlar/Insan_Arastirmalari_Kurumsal_Degerlendirme_Kurulu, accessed at March 2016.

105. PsychCentral, “Obsessive-Compulsive Disorder (OCD) Screening Quiz”, http://psychcentral.com/quizzes/ocdquiz.htm, accessed at March 2016.

106. Goodman, W. K., “The Yale–Brown Obsessive–Compulsive Scale”, *Arch Gen Psychiatry*, Vol. 46, pp. 1006–1011, 1989.

107. Provos, N., D. Mazieres and J. S. Talan, “A Future-Adaptable Password Scheme”, *Proceedings of the 1999 USENIX Annual Technical Conference*, pp. 81–92, 1999.

108. İrfan, B., A. A. Salah and H. L. Akın, “Yerleştirme Puanlama Anketi”, http://robot.cmpe.boun.edu.tr/~bahar/annotation/annotation_form.php, accessed at March 2016.

109. Yarpiz, “Particle Swarm Optimization in MATLAB”, http://yarpiz.com/50/ypea102-particle-swarm-optimization, accessed at March 2016.

110. İrfan, B. and H. L. Akın, “Analysis of Optimization Methods for Placement Planning”,

http://robot.cmpe.boun.edu.tr/~bahar/placementExperimentTables.html, accessed at March 2016.

111. İrfan, B. and H. L. Akın, "Placement and Manipulation Planning Experiments", http://robot.cmpe.boun.edu.tr/~bahar/placementExperiments.html, accessed at March 2016.

112. İrfan, B. and H. L. Akın, "Placement and Manipulation Planning Experiments in Gazebo", http://robot.cmpe.boun.edu.tr/~bahar/gazeboExperiments.html, accessed at March 2016.

APPENDIX A: DERIVATION OF ERROR FUNCTIONS FOR THE KINEMATICS

In order to eliminate the error in position caused by the 6 DOF kinematics calculations from *ROS MoveIt!* library for our 5 DOF arm, we used error functions, which depend on the target (x, y, z) coordinates. Therefore, we defined four fixed z positions for the gripper relative to the table. These z positions correspond to the z coordinate where the arm is

- Grasping the mug from side (6 cm above table) (z_L)
- Holding the mug above the table in the side grasp (7 cm above the highest mug $\approx$ 18 cm above table) (z_U)
- Turning the mug upside down above the table (7 cm above the highest mug $\approx$ 18 cm above table) (z_{Uud})
- Placing the mug on the tray upside down (6 cm above table) (z_{Lud})

where $z_L, z_U, z_{Lud}, z_{Uud}$ denote lower z, upper z, lower upside down z and upper upside down z coordinate, respectively. For each of these z positions, we have fitted error functions for target values in x, y, and z coordinates of the end effector (gripper). The fitted functions are 2-variable (target x (x_T) and y coordinates (y_T)) quadratic equations, which were the best fit models that produced minimum root mean square error (RMSE) with the maximum RMSE equal to 0.001519. Figure A.1 shows the error functions derived for x, y and z in the side grasp and Figure A.2 shows the error functions for upside-down side grasps. In Figure A.1, $x_L, y_L, x_U, y_U, z_L, z_U$ represent the real end effector coordinates gathered for the lower and upper z values, z_L and z_U. Likewise, in Figure A.2, $x_{Lud}, y_{Lud}, x_{Uud}, y_{Uud}, z_{Lud}, z_{Uud}$ represent the real end effector coordinates gathered for the lower and upper z values for upside down grasps, z_{Lud} and z_{Uud}.

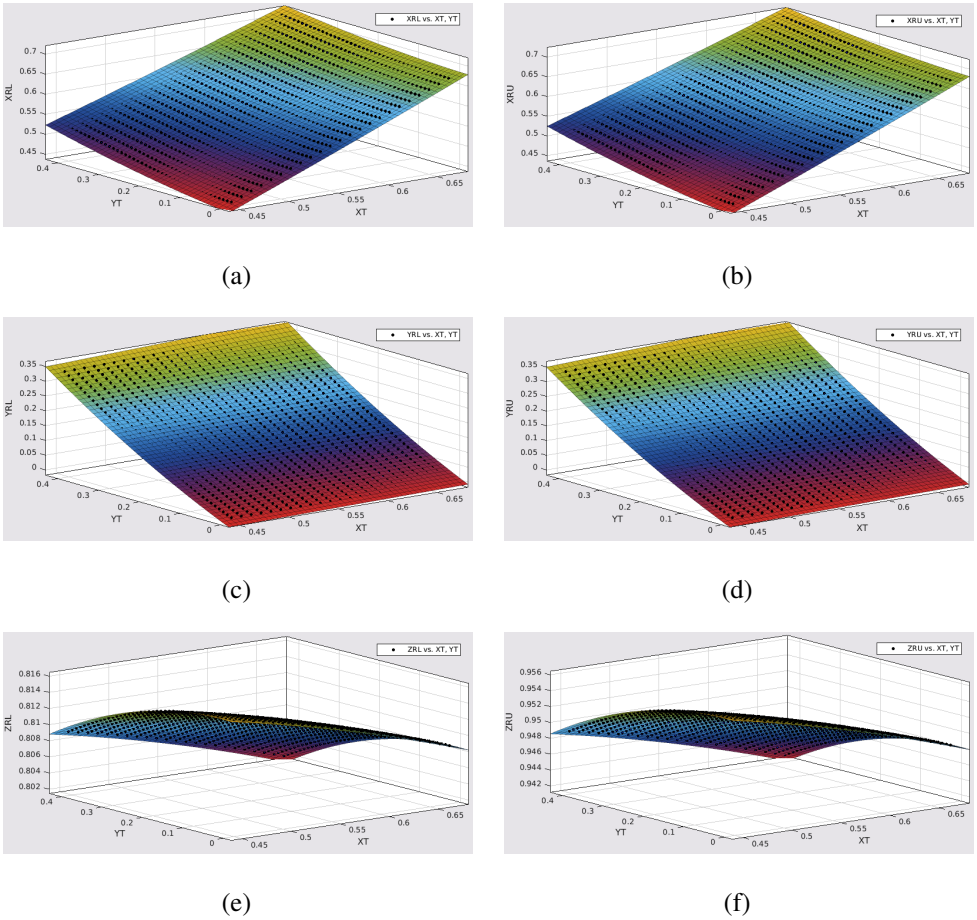
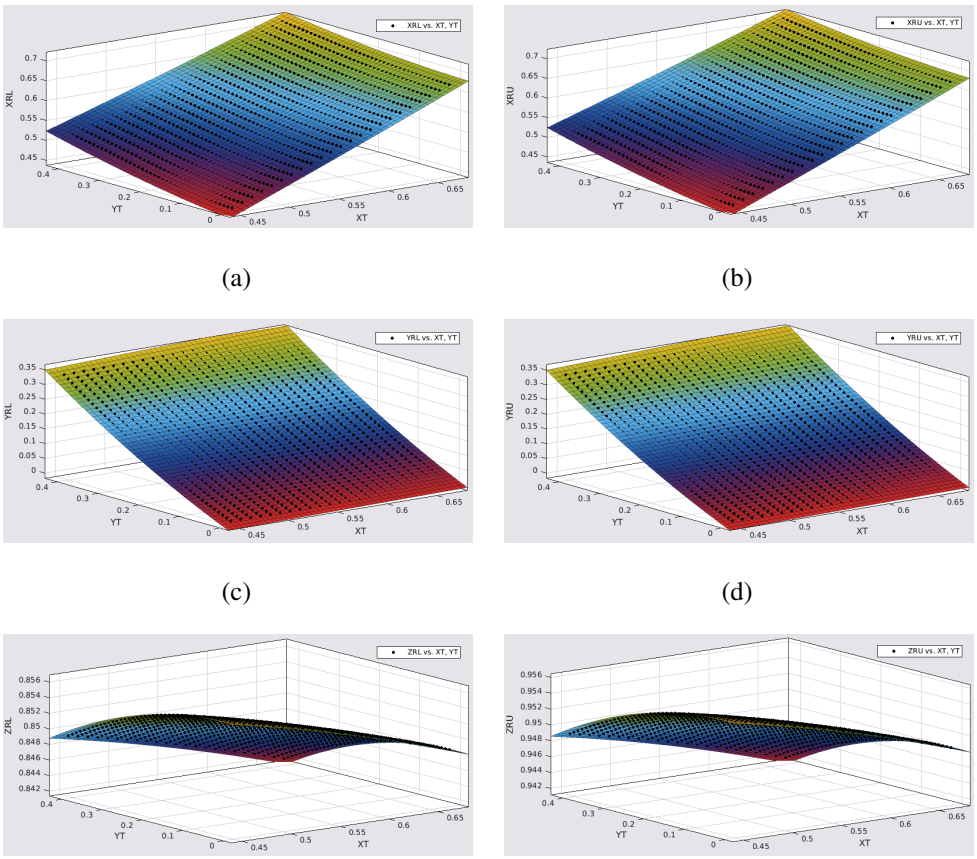


Figure A.1. Kinematic position error function graphs for side grasps: Functions of (a)x_L, (b)x_U, (c)y_L, (d)y_U, (e)z_L, (f)z_U.

Although the coefficients of the parameters for the error functions differ slightly from each other at z_L and z_U coordinates, we used them separately since we wanted to find the closest fit to decrease the difference between the actual target position and the real target position. The target pose is found by using the three functions given the actual target positions. The average fit functions for z coordinates outside our fixed positions are

$$\hat{x} = x + (x - (0.034652 + 0.84905x + 0.3283|y| + 0.146775x^2 - 0.437175x|y| + 0.1592y^2)) \quad \text{(A.1)}$$



(a) (b)

(c) (d)

(e) (f)

Figure A.2. Kinematic position error function graphs for upside down grasps: Functions of (a)x_{Lud}, (b)x_{Uud}, (c)y_{Lud}, (d)y_{Uud}, (e)z_{Lud}, (f)z_{Uud}.

$$\hat{y} = y \pm (|y| - (-0.037373 + 0.0927725x + 0.5603|y| - 0.04269x^2 + 0.1567x|y| + 0.450425y^2)) \quad \text{(A.2)}$$

$$\hat{z} = z + (z - (z + 0.007373x - 0.001207|y| - 0.03678x^2 + 0.002317x|y| - 0.040748y^2)) \quad \text{(A.3)}$$

where $\hat{x}$, $\hat{y}$ and $\hat{z}$ are the calculated target positions to give to the kinematics in order to obtain actual target positions in the vertical direction to the robot (x), in the horizontal direction (y), and in the upwards direction (z). The error function is symmetric around the y axes, hence an absolute value of y is used in the equations. The $\pm$ sign represents the situation where if y is negative the difference is subtracted, and if y is positive the difference is added. x and z always have positive values since the frame is relative to the coordinate frame of the base of the robot and the arm is in front of and above the base. The maximum error between the actual target position and the real position of the end effector was 17.7 cm before the correction, and it became 3.2 cm after the correction using the error functions.

APPENDIX B: USER STUDY SURVEY QUESTIONNAIRE

Bulaşık makinesine bulaşık yerleştirmede sizin için hangi kriterlerin daha önemli olduğunu anlayabilmemiz için bu anketi hazırladık. Bu anketin sonuçları uluslarası bir robotik konferansında kullanılmak üzere analiz edilecektir. Herhangi bir ticari amaç gütmemektedir. Bireysel bilgileriniz herhangi bir kurumla veya kuruluşla paylaşılmayacaktır. Anketimize katılımınızı önemle rica ederiz.

1. Cinsiyetiniz

Kadın

Erkek

2. Yaşınız

<18

18-25

26-35

36-50

51-65

>65

3. Çalışıyor musunuz?

Evet

Hayır

4. Öğrenci misiniz?

Evet

Hayır

5. Bulaşık makineniz var mı?

Evet

Hayır

6. Bulaşık makinenizi kullanım sıklığınız nedir?

İki haftada bir veya daha az

Haftada bir

Haftada 2-3 kere

Haftada 3'ten fazla

Figure B.1. User study questionnaire.

7. Kirli bardaklarınız tezgahta yığılı ise, bulaşık makinenizin üst gözünü doldurmanız sizce kaç dakika alır?

2 dakikadan az

3-5 dakika

6-10 dakika

10 dakikadan fazla

8. Bulaşık makinenizi yerleştirmede bir robot yardımcınız olsa, sizin için aşağıdaki kriterlerin önem sırası nasıl olurdu? 1 en önemli kriter olacak şekilde, 1'den 4'e kadar sıralayınız. Sizin için önemsiz olduğunu düşündüğünüz kriteri önemsiz olarak işaretleyiniz.

	Maksimum kapasiteyi kullanabilmesi	Önemsiz
	Kısa sürede yerleştirilmesi	Önemsiz
	Harcanılan elektrik enerjisinin az olması	Önemsiz
	Bardakların boyutlarına göre aynı gözde farklı yerlere koyulması	Önemsiz

Bitti

Figure B.1. User study questionnaire. (cont.)

APPENDIX C: OBSERVATIONAL STUDY

KATILIMCI BİLGİ ve ONAM FORMU

Araştırmayı destekleyen kurum: Boğaziçi Üniversitesi

Araştırmanın adı: Bir robotun bulaşık makinesini doldurması için bardak taşıma ve yerleştirme planlaması

Araştırmacının adı: Bahar İrfan

Adresi: Boğaziçi Üniversitesi, Bilgisayar Mühendisliği, Bebek, İstanbul 34342

E-mail adresi: bahar.irfan@boun.edu.tr

Telefonu: 0212 359 7095

Proje konusu: Bulaşıkları bulaşık makinesine yerleştirmek zaman alır ve birkaç günde bir tekrar edilmesi gerekir. Çalışmamız bir bardakları bulaşık makinesine otomatik yerleştiren robotik bir sistem yapmayı amaçlamaktadır. Bu deneydeki hedefimiz insanların bulaşık makinelerine bardak yerleştirirken neye önem verdiklerini anlamak, yerleştirmedeki karar mekanizmalarını incelemek ve yerleştirme sırasında elde edilen verileri inceleyip robotik projemize uygulamaktır.

Onam: Bulaşık makinesine bardak yerleştirirken önem verilen kriterleri bulmak üzerine yapmak istediğimiz araştırmaya katılmaya sizi davet ediyoruz. Araştırmaya katılmayı kabul ettiğiniz takdirde size deneyimizin öncesinde ve sonrasında vereceğimiz anketleri doldurmanızı rica ediyoruz. İlk anketimiz 11 sorudan oluşmaktadır ve bulaşık makinenize bulaşık yerleştirmede sizin için hangi kriterlerin daha önemli olduğunu anlamımıza yardımcı olacaktır. Bu anketi doldurmak en fazla iki dakikanızı alacaktır. Diğer anketimiz ise 15 soru içermektedir, ve en fazla beş dakikanızı alacaktır. İkinci yapacağımız anket bulaşıkları dizmedeki düzenlilik kriterinin sizin için neden önemli olduğunu anlamamızı sağlayacaktır.

Deneyimiz için size verdiğimiz farklı boyutlardaki iki tepsiye masanın üzerindeki iki farklı boyuttaki bardakları yerleştirmenizi istiyoruz. İki tepsi için de üç farklı bardak grubuyla yerleştirme yapılacaktır. Bu deneyler sırasında hareketlerinizi gözlemleyeceğiz, yerleştirme sürenizi ölçeceğiz ve deney sonucunda yerleştirdiğiniz bardakların resmini çekeceğiz.

Ekteki formda istenen bilgileri de sağlamanızı rica ediyoruz. İsminiz ve bu bilgiler tamamen gizli tutulacaktır. Çalışmaya katılmanız tamamen isteğe bağlıdır. Sizden ücret talep etmiyoruz ve size herhangi bir ödeme yapmayacağız. Bu deneyde elde ettiğimiz veriler ileride başka çalışmalar için de kullanılabilir. Katıldığınız takdirde çalışmanın herhangi bir aşamasında herhangi bir sebep göstermeden onayınızı çekme hakkına da sahipsiniz.

Yapmak istediğimiz araştırmanın size risk getirmesi beklenmemektedir. Araştırma sonucunda katkıda bulunarak insan sağlığına yarar sağlamasını beklemekteyiz. Bu formu imzalamadan önce, çalışmayla ilgili sorularınız varsa lütfen sorun. Daha sonra sorunuz olursa, Bahar İrfan'a da (Telefon: 0212 359 7095) sorabilirsiniz.

Adres ve telefon numaranız değişirse, bize haber vermenizi rica ederiz.

Figure C.1. Participant information and consent form.

Ben, ..., yukarıdaki metni okudum ve katılmam istenen çalışmanın kapsamını ve amacını, gönüllü olarak üzerime düşen sorumlulukları tamamen anladım. Çalışma hakkında soru sorma imkanı buldum. Bu çalışmayı istediğim zaman ve herhangi bir neden belirtmek zorunda kalmadan bırakabileceğimi ve bıraktığım takdirde herhangi bir olumsuzluk ile karşılaşmayacağımı anladım.

Bu koşullarda söz konusu araştırmaya kendi isteğimle, hiçbir baskı ve zorlama olmaksızın katılmayı kabul ediyorum.

Formun bir örneğini aldım / almak istemiyorum.

Katılımcının Adı-Soyadı:.....

İmzası:.....

Adresi (varsa Telefon No, Faks No):.....

.....

Tarih (gün/ay/yıl):...../...../.....

Varsa Katılımcının Vasisinin Adı-

Soyadı:.....

İmzası:.....

Tarih (gün/ay/yıl):...../...../.....

Araştırmacının Adı-Soyadı:.....

İmzası:.....

Tarih (gün/ay/yıl):...../...../.....

Figure C.1. Participant information and consent form. (cont.)

Anket 1

Bu anket bulaşık makinesine bulaşık yerleştirmede hangi kriterlerin daha önemli olduğunu anlayabilmek için hazırlanmıştır. Toplanan veriler bulaşık makinesini otomatik olarak dolduran bir mutfak robotunun tasarımında bilimsel girdi olarak kullanılacaktır. Bireysel bilgileriniz herhangi bir kurumla veya kuruluşla paylaşılmayacaktır. Anketimize katılımınızı önemle rica ederiz.

1. Cinsiyetiniz

- ◯ Kadın
- ◯ Erkek

2. Yaşınız

- ◯ <18
- ◯ 18-25
- ◯ 26-35
- ◯ 36-50
- ◯ 51-65
- ◯ >65

3. Çalışıyor musunuz?

- ◯ Evet
- ◯ Hayır

4. Öğrenci misiniz?

- ◯ Evet
- ◯ Hayır

5. Bulaşık makineniz var mı?

- ◯ Evet
- ◯ Hayır

Figure C.2. First questionnaire in the observational study.

6. Varsa bulaşık makinenizi siz mi dolduruyorsunuz?

- ◯ Evet
- ◯ Hayır
- ◯ Geçerli değil

7. Varsa bulaşık makinenizi siz mi boşaltıyorsunuz?

- ◯ Evet
- ◯ Hayır
- ◯ Geçerli değil

8. Bulaşık makinenizi kullanım sıklığınız nedir?

- ◯ İki haftada bir veya daha az
- ◯ Haftada bir
- ◯ Haftada 2-3 kere
- ◯ Haftada 3'ten fazla
- ◯ Geçerli değil

9. Kirli bardaklarınız tezgahta yığılı ise, bulaşık makinenizin üst gözünü doldurmanız sizce kaç dakika alır?

- ◯ 2 dakikadan az
- ◯ 3-5 dakika
- ◯ 6-10 dakika
- ◯ 10 dakikadan fazla

Figure C.2. First questionnaire in the observational study. (cont.)

10. Bulaşık makinenizi yerleştirmede bir robot yardımcınız olsa, sizin için aşağıdaki kriterlerin önem sırası nasıl olurdu? *1 en önemli kriter olacak şekilde, 1'den 4'e kadar sıralayınız. Sizin için önemsiz olduğunu düşündüğünüz kriteri önemsiz olarak işaretleyiniz.*

	Maksimum kapasiteyi kullanabilmesi	☐ Önemsiz
	Kısa sürede yerleştirilmesi	☐ Önemsiz
	Harcanılan elektrik enerjisinin az olması	☐ Önemsiz
	Bardakların boyutlarına göre dizilmesi (aynı boyuttaki bardakların arka arkaya veya yan yana dizilmesi)	☐ Önemsiz

11. Bardakların boyutlarına göre dizilmesi kriteri neden sizin için önemli?*Önemsiz seçeneğini işaretlediyseniz, bu soruyu boş bırakın.*

Figure C.2. First questionnaire in the observational study. (cont.)

Anket 2

Bu anket bulaşık makinesine bulaşıkları dizmedeki düzenlilik kriterinin neden önemli olduğunu anlayabilmemiz için hazırlanmıştır. Toplanan veriler bulaşık makinesini otomatik olarak dolduran bir mutfak robotunun tasarımında bilimsel girdi olarak kullanılacaktır. Bireysel bilgileriniz herhangi bir kurumla veya kuruluşla paylaşılmayacaktır. Anketimize katılımınızı önemle rica ederiz.

1. Toz, bakteri, kimyasal, radyasyon veya ciddi bir hastalık kapma hakkında sürekli olarak endişe duyar mısınız?

◯ Evet

◯ Hayır

2. Nesneleri düzgün bir sırada tutma veya yerleştirme konusunda aşırı titiz misiniz?

◯ Evet

◯ Hayır

3. Yangın, hırsızlık, sel gibi kötü olayların olmasından sık sık endişe duyar mısınız?

◯ Evet

◯ Hayır

4. Değer verdiğiniz bir nesneyi kaybetmekten çok endişe duyar mısınız?

◯ Evet

◯ Hayır

5. İstenmeyen davranışlarda bulunmaktan veya mantıksız dürtülere göre hareket etmekten hiç endişelendiniz mi? Örneğin, sevdiğiniz bir insana fiziksel olarak zarar vermek, trafiğin tersi yöne doğru araba sürmek gibi.

◯ Evet

◯ Hayır

6. Çamaşır yıkama, ortalığı temizleme, bakım yapma gibi işleri tekrar tekrar yapma gereği duyar mısınız?

◯ Evet

◯ Hayır

Figure C.3. Obsessive-Compulsive Scale test.

7. Işıkları, musluğu, ocağı, ütüyü veya kapı kilidini kapattığınızı kontrol etmeyi ihtiyaç duyar mısınız?

- ◯ Evet
- ◯ Hayır

8. Saymak, düzenlemek ve eşitlemek sık yaptığınız davranışlar mıdır?

- ◯ Evet
- ◯ Hayır

9. Rutin olarak yapılan hareketleri belli bir kere arka arkaya tekrarlama isteği duyar mısınız? Örneğin, sigarayı yakma, musluğu kapama, bir yere dokunma gibi.

- ◯ Evet
- ◯ Hayır

10. Yazdıklarınızın tekrar tekrar üzerinden geçme, veya mektuplarınızı postalamadan önce tekrar açıp kontrol etme eğiliminiz olur mu?

- ◯ Evet
- ◯ Hayır

11. Genel olarak, günde ortalama ne kadar bu tarz düşüncelere sahip olursunuz veya bu davranışlarda bulunursunuz?

- ◯ Hiç
- ◯ Az (1 saatten az)
- ◯ Orta (1-3 saat)
- ◯ Çok (3-8 saat)
- ◯ Aşırı (8 saatten fazla)

12. Sizi ne kadar rahatsız ediyorlar?

- ◯ Hiç
- ◯ Az
- ◯ Orta
- ◯ Çok
- ◯ Aşırı

Figure C.3. Obsessive-Compulsive Scale test. (cont.)

13. Bu düşünceler üzerinde ne kadar kontrolünüz var?

- ◯ Tam
- ◯ Çok
- ◯ Orta
- ◯ Az
- ◯ Hiç

14. Bir şey yapmaktan, bir yere gitmekten veya biriyle olmaktan ne ölçüde kaçınmanıza neden oluyorlar?

- ◯ Hiç
- ◯ Az
- ◯ Orta
- ◯ Sürekli ve çok
- ◯ Aşırı

15. Okulunuzu, işinizi, sosyal veya aile ilişkilerinizi ne kadar etkiliyorlar?

- ◯ Hiç
- ◯ Az
- ◯ Orta
- ◯ Çok
- ◯ Aşırı

Figure C.3. Obsessive-Compulsive Scale test. (cont.)

APPENDIX D: ANNOTATION

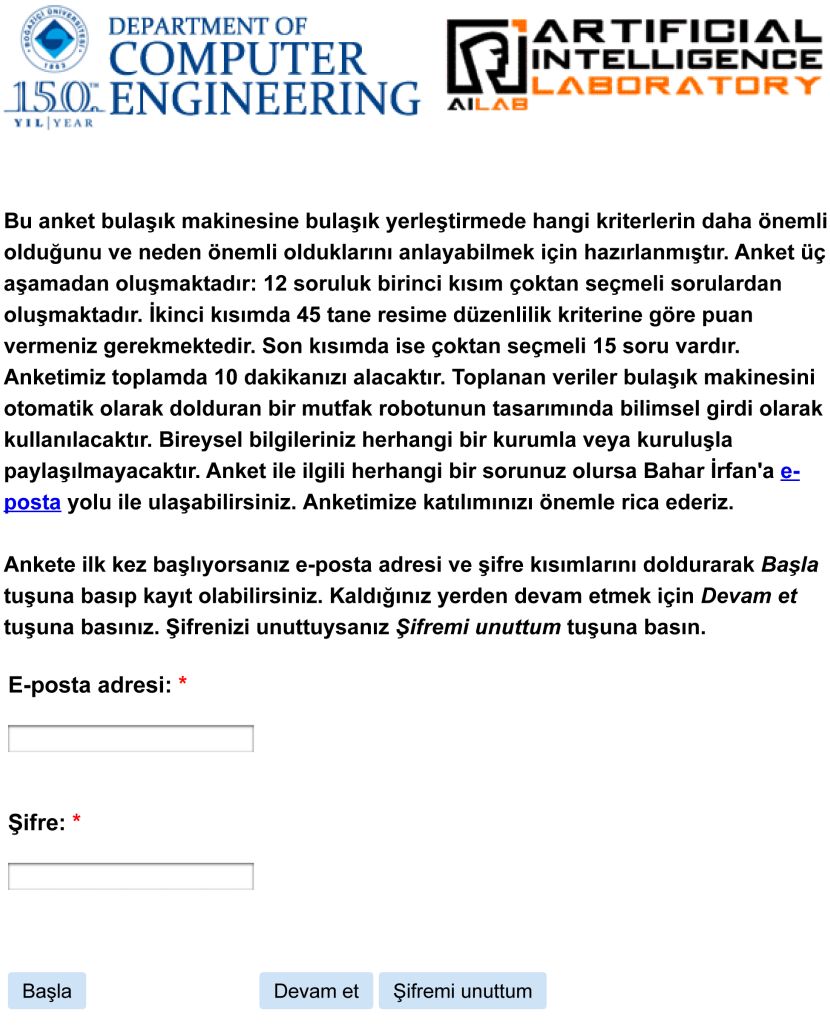


Figure D.1. Sign-up and log-in page for the annotation system.

Anket

1. Cinsiyetiniz

- ○ Kadın
- ○ Erkek

2. Yaşınız

- ○ < 18
- ○ 18-25
- ○ 26-35
- ○ 36-50
- ○ 51-65
- ○ > 65

3. Çalışıyor musunuz?

- ○ Evet
- ○ Hayır

4. Öğrenci misiniz?

- ○ Evet
- ○ Hayır

5. Bulaşık makineniz var mı?

- ○ Evet
- ○ Hayır

Figure D.2. First questionnaire for the annotation.

6. Varsa bulaşık makinenizi siz mi dolduruyorsunuz?

- ○ Evet
- ○ Hayır
- ○ Geçerli değil

7. Varsa bulaşık makinenizi siz mi boşaltıyorsunuz?

- ○ Evet
- ○ Hayır
- ○ Geçerli değil

8. Bulaşık makinenizi kullanım sıklığınız nedir?

- ○ İki haftada bir veya daha az
- ○ Haftada bir
- ○ Haftada 2-3 kere
- ○ Haftada 3'ten fazla
- ○ Geçerli değil

9. Kirli bardaklarınız tezgahta yığılı ise, bulaşık makinenizin üst gözünü doldurmanız sizce kaç dakika alır?

- ○ 2 dakikadan az
- ○ 3-5 dakika
- ○ 6-10 dakika
- ○ 10 dakikadan fazla

10. Bulaşık makinenizi yerleştirmede bir robot yardımcınız olsa, sizin için aşağıdaki kriterlerin önem sırası nasıl olurdu?
1 en önemli kriter olacak şekilde, 1'den 4'e kadar sıralayınız. Sizin için önemsiz olduğunu düşündüğünüz kriteri önemsiz olarak işaretleyiniz.

- ☐ Maksimum kapasiteyi kullanabilmesi
- ☐ Kısa sürede yerleştirilmesi
- ☐ Harcanılan elektrik enerjisinin az olması
- ☐ Bardakların boyutlarına göre dizilmesi (aynı boyuttaki bardakların arka arkaya veya yan yana dizilmesi)

Figure D.2. First questionnaire for the annotation. (cont.)

11. Robotunuzun bulaşık makinenizin üst gözünü en fazla kaç dakikada doldurmasını isterdiniz?

- 2 dakikadan az
- 3-5 dakika
- 6-10 dakika
- 10-15 dakika
- 15 dakikadan fazla

12. Bardakların boyutlarına göre dizilmesi kriteri neden sizin için önemli?
Önemsiz seçeneğini işaretlediyseniz, bu soruyu boş bırakın.

İleri

Figure D.2. First questionnaire for the annotation. (cont.)

Figure D.3. Annotation with Likert scale for a placement.

1. Toz, bakteri, kimyasal, radyasyon veya ciddi bir hastalık kapma hakkında sürekli olarak endişe duyar mısınız?

- ○ Evet
- ○ Hayır

2. Nesneleri düzgün bir sırada tutma veya yerleştirme konusunda aşırı titiz misiniz?

- ○ Evet
- ○ Hayır

3. Yangın, hırsızlık, sel gibi kötü olayların olmasından sık sık endişe duyar mısınız?

- ○ Evet
- ○ Hayır

4. Değer verdiğiniz bir nesneyi kaybetmekten çok endişe duyar mısınız?

- ○ Evet
- ○ Hayır

5. İstenmeyen davranışlarda bulunmaktan veya mantıksız dürtülere göre hareket etmekten hiç endişelendiniz mi? ***Örneğin, sevdiğiniz bir insana fiziksel olarak zarar vermek, trafiğin tersi yöne doğru araba sürmek gibi.***

- ○ Evet
- ○ Hayır

Figure D.4. Obsessive-Compulsive Scale test used in the annotation study.

6. Çamaşır yıkama, ortalığı temizleme, bakım yapma gibi işleri tekrar tekrar yapma gereği duyar mısınız?

- ○ Evet
- ○ Hayır

7. Işıkları, musluğu, ocağı, ütüyü veya kapı kilidini kapattığınızı kontrol etmeyi ihtiyaç duyar mısınız?

- ○ Evet
- ○ Hayır

8. Saymak, düzenlemek ve eşitlemek sık yaptığınız davranışlar mıdır?

- ○ Evet
- ○ Hayır

9. Rutin olarak yapılan hareketleri belli bir kere arka arkaya tekrarlama isteği duyar mısınız? *Örneğin, sigarayı yakma, musluğu kapama, bir yere dokunma gibi.*

- ○ Evet
- ○ Hayır

10. Yazdıklarınızın tekrar tekrar üzerinden geçme, veya mektuplarınızı postalamadan önce tekrar açıp kontrol etme eğiliminiz olur mu?

- ○ Evet
- ○ Hayır

11. Genel olarak, günde ortalama ne kadar bu tarz düşüncelere sahip olursunuz veya bu davranışlarda bulunursunuz?

- ○ Hiç
- ○ Az (1 saatten az)
- ○ Orta (1-3 saat)
- ○ Çok (3-8 saat)
- ○ Aşırı (8 saatten fazla)

Figure D.4. Obsessive-Compulsive Scale test used in the annotation study. (cont.)

12. Bu düşünceler veya davranışlar sizi ne kadar rahatsız ediyor?

- ○ Hiç
- ○ Az
- ○ Orta
- ○ Çok
- ○ Aşırı

13. Bu düşünceler veya davranışlar üzerinde ne kadar kontrolünüz var?

- ○ Tam
- ○ Çok
- ○ Orta
- ○ Az
- ○ Hiç

14. Bir şey yapmaktan, bir yere gitmekten veya biriyle olmaktan ne ölçüde kaçınmanıza neden oluyorlar?

- ○ Hiç
- ○ Az
- ○ Orta
- ○ Sürekli ve çok
- ○ Aşırı

15. Okulunuzu, işinizi, sosyal veya aile ilişkilerinizi ne kadar etkiliyorlar?

- ○ Hiç
- ○ Az
- ○ Orta
- ○ Çok
- ○ Aşırı

İleri

Figure D.4. Obsessive-Compulsive Scale test used in the annotation study. (cont.)

Anketimiz bitmiştir. Katıldığınız için teşekkür ederiz.

Figure D.5. Final page of the annotation system.