INTEGRATING AND EVALUATING PLANNING PRIMITIVES FOR ROBOT MANIPULATION TASKS IN WAREHOUSE LOGISTICS

by

ZACHARIAS PSARAKIS

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Kostas Bekris

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

April, 2018

ABSTRACT OF THE THESIS

Integrating and Evaluating Planning Primitives for Robot Manipulation Tasks in Warehouse Logistics

by Zacharias Psarakis

Thesis Director: Kostas Bekris

The Amazon Robotics Challenge was an event created by Amazon to bring robotics teams together and try to push forward the research on robotics automation related to warehouse logistics. The challenge consists of two tasks. The picking task requires a fully autonomous robotic system to remove target objects from a shelving unit into a tote. The stowing task requires placing objects from a tote into the shelving unit. The purpose of this thesis is to present state-of-the-art approaches related to motion, grasp and task planning, pose estimation solutions, and how different end-effector modalities can affect the performance of the autonomous system. Two grasp planning and three task planning approaches are evaluated and combined with a PRM* motion planner in the context of the Amazon Robotics Challenge tasks.

Acknowledgements

I would like to thank my supervisor, Kostas Bekris, who was amazing, wise and patient throughout our interactions the past 3 years. I would also like to thank my sister Konstantina Psaraki for being the awesomest sister in the world. My parents Manolis Psarakis and Dionisia Tzaki for everything they taught me and provided the past 29 years. My colleagues from PRACSYS lab, Thanasis Krontiris, Hristiyan Kourtev, Shaojun Zhu, Colin Rennie, Zakary Littlefield, Andrew Dobson (Chuples), Rahul Shome, Andrew Kimmel, Chaitanya Mitash for all their support. Finally my friends, Nikolas Giannopoulos, Dionisis Kalogeropoulos, Dimitris Chaidas, Georgia Anastopoulou, Thanasis Krontiris, Varvara Kalokyri, Georgios Chantzialexiou, Christos Mitropoulos, Efthimis Mavris, Tasos Dimas, Georgios Tsilomelekis, Chrisalenia Koumpouzi, Ted Tsoulos, Matina Kalontzi, Anastasia Papadimitriou, Vilelmini Kalampratsidou and last but not least Hristiyan Kourtev for all the fun, late nights and beautiful moments.

Dedication

I dedicate this to Fw

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Warehouse Automation

The Amazon Robotics Challenge (Fig. 1.1) triggered a trend towards building a fast, robust and efficient system that could solve the problem of warehouse picking and stowing. Many labs in the US as well as around the world [1], [2], [3] focused their resources into building such a system and competing with each other. The acquisition of Kiva Systems by Amazon back in 2012 and the introduction of the Kiva bots and Kiva pods [4] in Amazon's warehouses was the first step towards a fully automated warehouse.

Figure 1.1: Amazon Robotics Challenge.[1]

Amazon's warehouses are on the top of the list of the most automated warehouses. Other than manually stowing the Kiva pods with the merchandise and removing the corresponding objects when an order comes in, the rest of the procedure is automated. Amazon owns 8 warehouses totalling 1.2 million square feet in size and it is obvious that managing such a big volume of merchandise can be challenging.

[1]Reprinted from "www.amazonrobotics.com"

1.1.1 Stowing

One would think that merchandise is stored in the Kiva pods depending on the type, size or category, sorted by price or in alphabetical order but that is not the case in an Amazon warehouse. Amazon uses the principal of chaotic storage. This means that there is no area for clothes or a department for appliances and electronics in a warehouse. A Kiva pod may contain multiple types of merchandise with different price ranges and characteristics. The properties of each product stored in a Kiva pod are irrelevant, and the only thing that matters is the unique bar-code. Note that apart from the products that enter an Amazon warehouse, each Kiva pod is also associated with a unique bar-code. Amazon warehouse employees scan incoming products and place them in an unoccupied space inside a Kiva pod registering the bar-code of the product as well as the bar-code of the pod. That way, at any given point, a central system knows in which pod a specific product can be found. Furthermore, the information of which pods are almost empty or full can be recovered by keeping track of how many products were stowed and removed from the pod. Considering the size of an Amazon's warehouse manually moving all the merchandise around and trying to find a place in a Kiva pod is a time consuming procedure (Fig. 1.3(a)). The solution to that problem was given by the Kiva bots.

Figure 1.2: The Kiva Bot.[2]

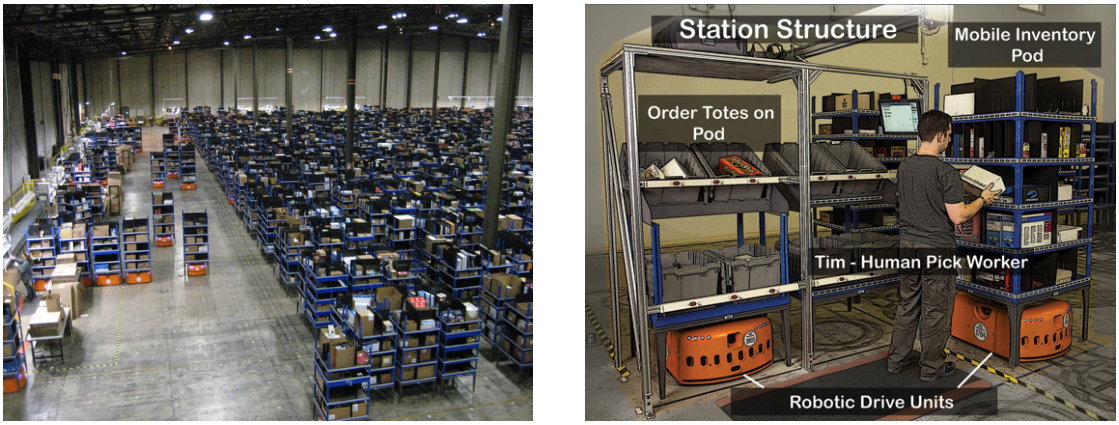
Instead of moving the incoming products inside the warehouse and placing them in the pods, using the Kiva bots, the most empty storage units are moved at the entrance of the

[2]Reprinted from "https://www.bloomberg.com/news/articles/2015-05-28/robot-with-a-human-grasp-is-amazon-s-challenge-to-students"

warehouse where the new products are stowed (Fig. 1.2). After the stowing process the Kiva bot moves the pod back to the warehouse, updating its final placement. It is worth mentioning that each pod does not have a specific pre-defined position, but it is placed closer to the entrance or the worker stations of the warehouse depending on how full it is.

1.1.2 Picking

When a customer's order comes in, Kiva bots choose the most optimal pods, considering the distance that they have to travel and the number of different pods needed to fulfil the order, and transfer them to the worker stations where an Amazon employee is waiting to pick the products out of the Kiva pod (Fig. 1.3(b)), place them in boxes and labelling them for shipping to the customer. The final position of the Kiva pod will be decided based on how many products are left inside: fewer products means that the pod will be placed closer to the entrance of the warehouse, reducing the time required to transfer it there for re-loading. An almost full pod will remain close to the worker stations since it is more possible that it will contain a product included in another order.



(a) The interior of an Amazon warehouse.

(b) Human worker in an Amazon warehouse.

Figure 1.3: Amazon warehouse automation: Kiva bots move the pods to the human workers.[3]

[3](a) Reprinted from "https://www.roboticsbusinessreview.com/supply-chain/amazon_warehouse_ demand_devours_robots_and_workers/". (b) Reprinted from "https://www.firstpost.com/tech/news-analysis /amazon-rolls-out-more-than-15-000-kiva-robots- for-holiday-season-onslaught-3659875.html"

1.2 The Kiva Pod

All the objects in an Amazon warehouse are stored in Kiva pods. The Kiva Pod (Fig. 1.4) has 12 bins, each of which is about 20(width)x25(height)x40(depth) cm size. The two middle rows of bins have 20cm height. Given the depth, it is difficult to reach objects deep

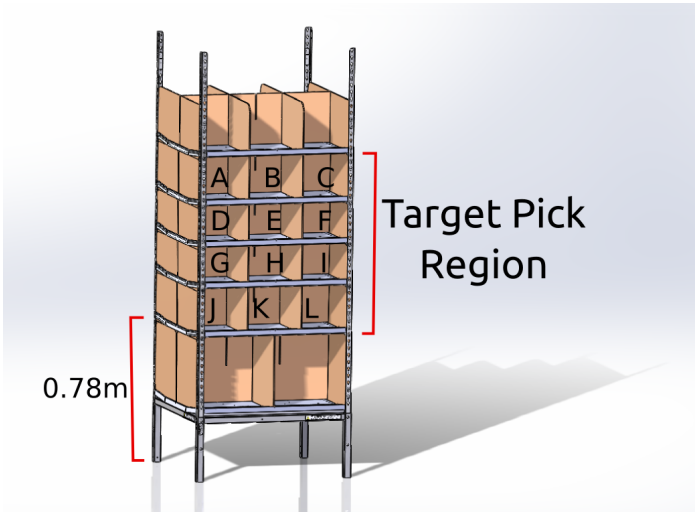


Figure 1.4: The Kiva Pod.

inside them without extending the robot's arms. This is one of the reasons that the choice of the end-effectors that will be used affects the overall system's performance and success rate. For this reason our vacuum grippers share some common properties, in particular they have a long shaft and 1-DOF joint at the tip, which allows the vacuum side to rotate up to 90 degrees.

1.3 The Amazon Robotics Challenge

In this section we will describe the history of the Amazon Robotics Challenge. How it begun, what changed over the years, what are the challenges and what were the important innovation the teams contributed to the robotics community. The purpose of the challenge was to motivate research groups to provide a fully autonomous system integrating state of the art object recognition, motion, task and grasp planning, as well as effective end-effector solutions that would perform well in an industrial environment.

1.3.1 The 2015 Amazon Picking Challenge

In 2014, Amazon announced the 1^{st} Amazon Picking Challenge competition which took place in May 2015 as part of the International Conference on Robotics and Automation (ICRA) in Seattle, WA. The objective of the competition was to remove from a Kiva pod a set of objects with a fully autonomous robotic solution.

Picking Task

The objects was placed in the Kiva pod in such a way that the bins were distinguished in 3 categories:

- Single-item bins: At least two bins only contained one object. Both of these objects were picking targets.
- Double-item bins: At least two bins contained two objects. One object from each of these bins was a picking target.
- Multi-item bins: At least two bins contained three or more objects. One object from each of these bins was a picking target.

The teams were provided a JSON file that listed the objects that should be picked up. Each competing team had 20 minutes to remove 12 objects, out of the total of 24 objects, from the Kiva pod and place them inside an order bin. The task proved to be challenging as no team managed to complete the full challenge within the time limit. For each successful pick, the competing team gained 10, 15 or 20 points depending on how many objects were in the bin that the target object was picked from. Some objects, considering their uniqueness in shape and/or texture awarded 1-3 bonus points. Teams were penalized with 12 points if they removed a non-target object from the Kiva pod, 5 points if they damaged any object or packaging and 3 points if they dropped an object from a height more than 30cm. Successful was considered a pick that removed a target object from the Kiva pod, transferred it without damaging the package or the object and dropped it in the order bin from a height less than 30cm.

Figure 1.5: The RBO team's robot.[4]

Winning Team

Team RBO from the Technical University of Berlin [5] won the challenge with 10 out of 12 successful picks and 148 points leaving Team MIT in the second place with 7 picks and 88 points.

The winning team chose to use a simple vacuum cleaner shaft as an end-effector mounted on a single Barrett arm and a mobile base XR4000 (see Fig. 1.5). For pose estimation purposes they mounted a 3D imaging camera on the arm and for collision checking when moving the mobile base they used a laser sensor. Instead of using motion planning algorithms to come up with collision free plans, they employed pressure and force-torque sensors and all the motions were based on control and sensor feedback.

1.3.2 The 2016 Amazon Picking Challenge

In 2016 the Amazon Picking Challenge additionally to the picking task, a stowing task was also introduced. Competing teams were required to pick 12 objects out of the Kiva pod and additionally move 12 different objects from a tote into a partially full Kiva pod. The set of objects used was significantly larger than the first year, consisting of a total of 46 objects

[4]Reprinted from "http://robotglobe.org/team-rbo-wins-the-amazon-picking-challenge/."

with large variation in size texture shape and weight.

Picking Task

For the picking task the objects were placed inside the Kiva pod so that:

- Each bin contained between one and ten objects.
- Overall the twelve bins contained a total of 50 objects. Notice that 4 objects were not unique.
- Each bin contained exactly one target object.
- The objects inside of each bin could be partially occluded or in contact with other objects, making the pose estimation more challenging.

In the beginning of the challenge, each team was provided with a JSON file containing the target objects. Each team was required to provide a task output file that contained the final configuration of the Kiva pod i.e. which objects are in which bin after the team's attempt. Competing teams had 15 minutes to pick the 12 target objects in the order bin. For each successful pick a team was awarded with 10 points if the target object was in a bin with 1 or 2 objects, 15 points if the bin contained 3 or 4 objects or 20 points if the bin contained 5 or more objects. Similarly to the 2015 Amazon Picking Challenge, points were deducted for removing non-target objects from the Kiva pod, damaging any object or the Kiva pod, dropping an object from a height more than 30cm. Additionally points were deducted if the task output file reported an incorrect final position for an object and if an object was protruding out of its bin by more than 5cm.

Stowing Task

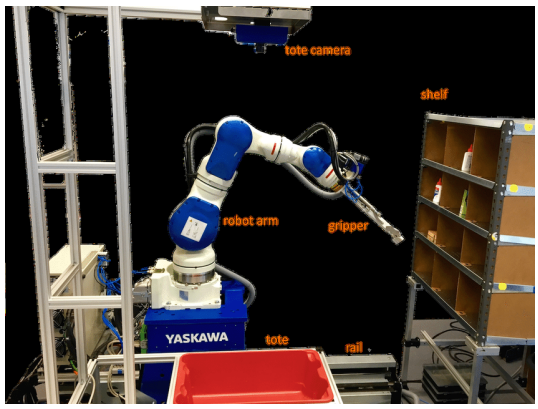
Regarding the stowing task each team had 15 minutes to remove 12 objects from a tote and place them inside the Kiva pod:

- 40 non-target objects were placed in the Kiva pod.
- Each bin contained between one and ten objects.

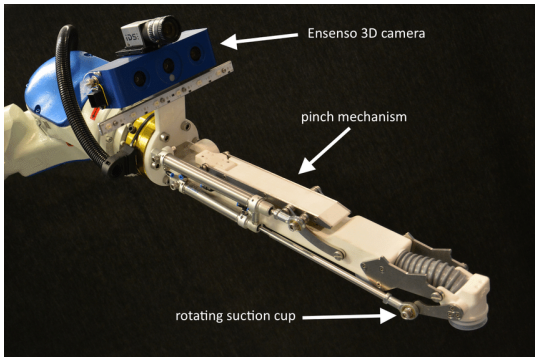
- The tote contained only the 12 target objects.
- The target objects were partially or completely occluded below other objects.

Winning Team

Team Delft from TU Delft Robotics Institute [6] won both challenges. Team Delft used a 7 DoF SIA20F Motoman robot arm by Yaskawa mounted on a rail which provided an extra DoF increasing the reachability of the robot (Fig. 1.6(a)). For pose estimation they used 2 industrial stereo cameras. One was mounted on the arm to scan the bin of the Kiva pod and the other over the tote to estimate the pose of the objects during the stowing task. Pose estimation consisted of 2 steps. For object recognition they used a deep learning neural network based on faster R-CNN and a step of global optimization for the initial estimation using super4PCS. Team Delft developed a custom hybrid end-effector featuring a suction cup with 1 DoF and a pinch mechanism to perform grasps on difficult to suck (Fig. 1.6(b)).



(a) Team Delft Robot.



(b) Team Delft custom hybrid end-effector.

Figure 1.6: Team delft won the 2016 Amazon Picking Challenge using a custom hybrid end-effector. [5]

Regarding motion planning, team Delft divided the planning problem into 2 sub-problems: motion outside the Kiva pod where the trajectories were pre-recorded since the environment was not dynamic, and motion inside the bins where the sense-plan-act approach was used.

[5] Reprinted from [6].

1.3.3 The 2017 Amazon Robotics Challenge

The Amazon Picking Challenge was renamed in 2017 into Amazon Robotics Challenge. The challenge focused again on picking and stowing tasks but also allowed teams to create their own storage systems. The challenge consisted of three phases. The first and second phases were similar to last year's picking and stowing tasks. In the final phase only the best teams from the first two phases competed and the task was to first stow 10 objects in the storage system and then pick them again and separate them in 3 distinctive groups. One major difference compared to the previous challenges was the set of objects. The competing teams were provided with a set of 40 objects prior to the challenge. At the day of the challenge though, a different set of objects was used. Half of the objects were included in the initial set, but the other half were new objects provided to the team around 30 minutes before the beginning of the event, challenging that way the pose estimation approaches for each team.

Custom Storage System

Teams were allowed to design their own storage system following certain specifications that their purpose were to maintain the intention and spirit of the competition to be the advancement of state-of-the-art robot manipulation and perception. A custom storage system:

- Must consist of between one and three objects that are physically distinct from the Robot and that could be reasonably carried by an Amazon Robotics mobile robot. These objects may be bolted down or otherwise secured in place for the Challenge.
- Must have at least 2 and no more than 10 distinct internal bins that are clearly labelled with letters A through J. All objects must be stored within these bins.
- The sum of the outer upright bounding box volumes of the object(s) must be no more than 95,000 cubic centimetres. All bins must be contained inside this bounding box. Outer bounding shapes such as cylinders or pyramids may also be considered.
- Must occupy an area of no more than 5,000 square centimetres, and its length, width, or height must be no longer than 125 centimetres. It may be placed on top of a table

or other supporting structure.

- Must be static and contain no motors or actuators. It may contain parts such as drawers that are moved by the robot, provided that these parts start and end the task inside the bounding box of the Storage System.
- Any sensors contained within the Storage System must be inexpensive, with a total price for all such sensors of less than $50.
- The same Storage System must be used of the Pick, Stow, and Final Round Tasks.

Picking Task

The introduction of the custom storage system changed the way the challenge started for each team. The total number of objects was 32 and each team had the opportunity to store the objects inside their storage system themselves. Note that 16 objects were part of the object set announced to the team prior to the challenge, while the other 16 were new objects.

- Before each team's attempt, judges handed the objects to the team one by one.
- Team members decided the final placements of each object in their storage system.
- The placements of all the objects had to be completed in less than 8 minutes.
- The team then had 15 minutes to complete 3 work orders: A 5 object order, a 3 object order and a 2 object order
- Each order had an associated cardboard box and target objects had to be packed in the correct box for the order.

The competing team was awarded 10 points for each successful placement of a target object inside the correct box plus an additional 10 points if the objects was one of the new ones. Furthermore, an additional point for every 5 seconds remaining on the clock after they completed the task and 10 points if the cardboard box could be closed without having to re-arrange the objects inside. Regarding point deductions, 15 points were deducted for

each non-target objects that was not in the storage system at the end of the task, 15 points for each target object that was not in the storage system, or in the correct box at the end of the task, 5 points for each target object that was dropped from a height more than 15cm, 5 points for each item that was protruding more than 2cm out of the storage system, and 5 and 20 points for minor and major damage on an object or a box respectively. An amnesty tote was also introduced. If a team grasped an object that could not be recognized by their sensing software they could place it in the amnesty tote and no points would be deducted for the specific object.

Stowing Task

For the stowing task:

- The competing teams were provided with a tote filled with 20 objects in an unstructured jumble.
- 10 of the objects were part of the initial object set, while the rest were new objects.
- The tote was placed in a position indicated by the team.
- The robot had 15 minutes to move all 20 objects in the storage system.
- At the end of the task, the final location of each object had to be reported.

For each object successfully stowed in the storage system the team was awarded 5 points plus 5 additional points if the object was not part of the initial object set. Furthermore, an additional point for every 5 seconds remaining on the clock after they completed the task given that at least 15 of the objects were reported at their actual final location. Similarly to the picking task, 15 points were deducted for each object not in the storage system or the tote at the end of the stowing task. Additionally, 5 points were deducted for each object in the storage system or tote with an incorrect final position, 5 points for each object dropped from a height greater than 15cm, 5 points for every object protruding more than 2cm out of the storage system and 5 and 20 points for minor and major damage on an object or a box respectively. The competing team could use the amnesty tote during the stowing task too, placing inside objects that could not be recognized.

Final Round

The top teams that gathered the most points during the picking and stowing task competed in a final round task that combined stowing and picking. A team had to reach a minimum of 100 points in either the pick or stow tasks and rank in the top 10 teams based on the sum of the points gathered in the 2 tasks to qualify to the final round.

- 2 totes containing 16 objects each were provided to each finalist team. Half of the objects in each tote were included in the initial object set, while the other half were new objects.
- Each team had to place the contents of the first tote in their storage system
- The robot had 30 minutes for the remainder of the final round divided between stowing and picking, however each team chose.
- The stowing task included placing all the objects of the second tote in the storage system. When the team declared that the robot has completed the stowing phase the time stopped and the remaining time of the 30 minutes was used for the picking task.
- The competing team was then provided with a work order similar to the picking phase. The 10 objects were chosen from the 32 initial objects regardless if the robot had succeeded in placing them in the storage system during the stowing task.
- Similarly to the first picking task, the order had to place objects in 3 different cardboard boxes.
- Points were awarder and subtracted using the same rubric as the initial stowing and picking tasks.

Winning Team

The winner of the 2017 Amazon Robotics Challenge was a custom-built, low-cost Cartesian robot, cartMan [7], built from scratch by the Australian Centre for Robotic Vision (ACRV) from Queensland University of Technology. Instead of a robotics arm, the winning team

(a) The Cartesian robot cartMan. (b) The custom storage system used by cartMan.

Figure 1.7: Australian Centre for Robotic Vision won the 2017 Amazon Robotics Challenge with a low-cost Cartesian robot. [6]

used a sliding mechanism that picked up products from above (Fig. 1.7(a)). The storage system designed for the purpose of the competition was 2 horizontally placed bins that were accessible from the top by cartMan (Fig. 1.7(b)). While other teams combine a vacuum end-effector and a pinching mechanism into a single tool, cartMan's designed allowed ACRV team to design an end-effector with 2 distinct tools selectable by a 180 degree rotation as shown in Fig. 1.8.

The teams choice of a pose estimation pipeline consisted of 2 key functions: *i)* segmentation and identification of the objects and *ii)* online generation of grasp points for each object. To achieve this they used RefineNet [8] a state-of-the-art semantic segmentation network for pixel-wise classification and a custom vision-based grasp approach (as opposed to a model-based one). For sensing, a RealSense SR300 RGB-D camera was mounted on the robot's wrist. Additionally, a second RealSense camera was mounted on the robot's frame to compensate for possible classification failures when using the wrist-mounted camera. An interesting approach was the way the grasp validation was performed. Scales were placed under the custom storage system that measured the weight before and after an attempted grasp. If the difference in weight was not the expected object's weight, the object was

[6](a) Reprinted from [7] (b) Reprinted from "http://www.i-programmer.info/news/169-robotics/10997-rob.html"

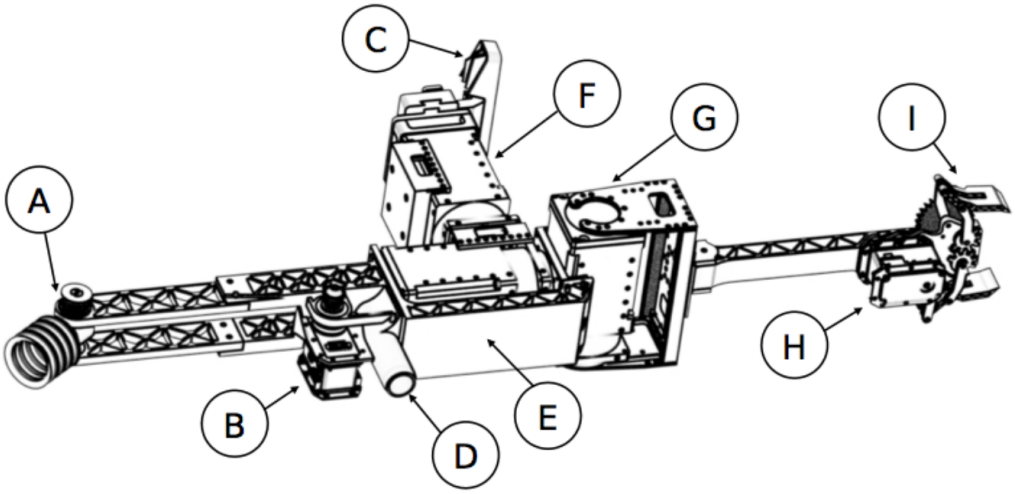


Figure 1.8: Rotating end-effector with 2 modalities. (A) Rotating suction cup (B) Suction gripper pitch servo (C) wrist-mounted RealSense camera (D) suction hose attachment (E) Roll motor (F) Yaw (tool-change) motor (G) Gripper pitch motor (H) Gripper servo (I) Parallel plate gripper. [7]

moved in front of the secondary camera to retry recognition. If there was no weight difference, the grasp was classified as non successful and the robot retried grasping the same or a different object. The team reported that the total cost of the robot, the storage system and the sensors used was approximately $24000, making it the most inexpensive setup in the challenge.

1.4 PRACSYS Group Participates in the Amazon Picking Challenge

R U Pracsys team from Rutgers University competed in 2015 Amazon Picking Challenge [9] and finished seventh. Following our participation in the competition, we evaluated the proposed system and our choice of end-effectors [10] [11]. Although the majority of the teams competing in the Amazon Challenges prefer using vacuum solutions, the results presented in our research show that there are cases that an underactuated 3-finger end-effector out performs the vacuum solution. This observation raises discussion about the need to have different end-effector modalities in order to achieve robust autonomy in an

[7] Reprinted from [7]

unknown industrial environment. Furthermore, it points out that a hybrid end-effector solution could help progress towards a fully automated robotic system.

1.5 Overview Of The Thesis

In Chapter 2 we will describe the hardware setup consisting of the manipulator, the end-effectors and the vacuum sources used. In Chapter 3 The software components of the system will be presented. More specifically, the task planner, the pose estimation software, the grasp planner and the motion planner will be described in detail along with the platform created to evaluate the system. In Chapter 4, the results of preliminary experiments will be presented that aim in identifying the bottlenecks of our initial approach and the difference in performance between end-effectors with different modalities. In Chapter 5 our proposed solution for the Amazon Robotics Challenge is evaluated. Finally in Chapter 6 we will propose improvements in order to further optimize the hardware and software components of the system under evaluation. All the charts regarding the performance of the system under review can be found in Appendix A.

Chapter 2

Hardware Setup And Gripper Design

2.1 Manipulator

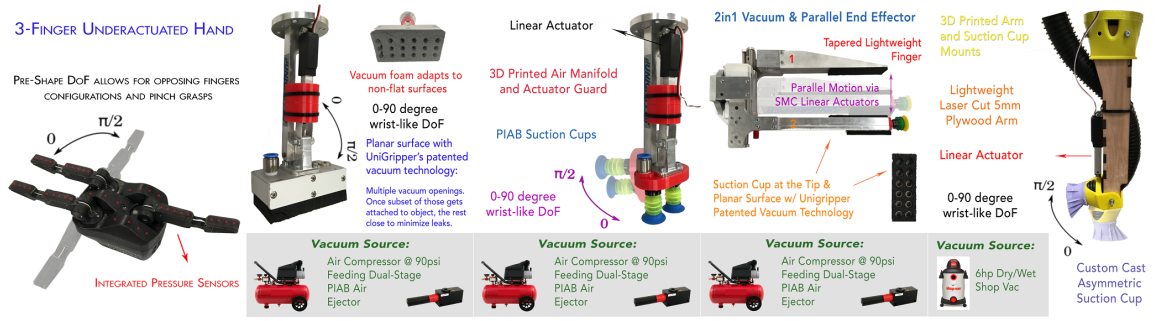
Figure 2.1: Hardware setup evaluated for warehouse picking.

The considered hardware setup consists of a dual-arm Yaskawa Motoman SDA10F. This system (see Fig. 2.1) is a dual arm robot, where each arm has 7 DoFs. The robot has one additional DoF at its torso, which can rotate by 180^o degrees in each direction. This

capability allows the robot to reach all twelve bins of the Kiva pod with both arms.

2.2 End-effectors

One important choice that has to be made is the type of end-effector that is going to be used. The following alternative end-effectors were tested: a) a RightHand Robotics ReFlex TakkTile Hand and a few custom-built grippers, b) a vacuum gripper designed together with UniGripper using their suction technology, which has 1 wrist-like DOF, c) an alternative 1-DOF vacuum gripper using suction cups, d) a hybrid parallel/vacuum gripper that also utilizes UniGripper's technology and e) a high-flow 1-DOF vacuum gripper.



Pros:
- Rubber fingers provide good friction
- Strong servo motors for heavy items (<5lb)
- Ability to execute caging grasps
- Built in pressure sensors allow for easy grasp validation

Cons:
- Even though compact relative to other similar hands, it is still large
- Hard to plan for
- Pre-generating grasps is time consuming

Pros:
- Wrist-like DoF allows approaching object from the front and from the top
- Maintains grasp even if a large number of the openings are unconcluded by the object (vacuum leak)

Cons:
- The presence of vacuum foam and the low flow of UniGripper's vacuum technology required the gripper to be pressed against the object in order to attach itself. Objects grasped from the front can easily topple over

Pros:
- DoF allows approaching object from the front and from the top
- 2 suction cups vs array of openings greatly increases the suction force produced (objects attach to end-effector much easier)
- More compact compared to UniGripper's vacuum solution

Cons:
- More accurate planning is required as good seal is needed at Both suction cups. Leaving one open is a very large leak.

Pros:
- Suction cup at the tip allows reaching in tight spaces and attaching to lighter objects securely
- UniGripper planar surface allows grasping flat objects
- Relatively thin Parallel fingers allow for reaching deep inside bin.
- Strong parallel motors allow grasping of heavy objects

Cons:
- Large size due to UniGripper vacuum technology, long fingers and need to pick heavy objects (large motors required)

Pros:
- Much higher air flow can tolerate very large leaks and works with uneven and porous objects, requiring less precise planning
- Asymmetric cup provides secure horizontal grasps while keeping overall cup diameter small
- DoF allows approaching object from the front and from the top

Cons:
- High flow requires a thick hose and large suction cup, which limits the minimum size of gripper achievable

Figure 2.2: Gripper comparison - from left to right: (a) The ReFlex Hand, (b) Unigripper solution, (c) Unigripper solution with suction cups, (d) Hybrid Unigripper end-effector, (e) Custom high-flow vacuum gripper

2.2.1 The ReFlex TakkTile Hand

The "ReFlex"hand[1], shown in Fig. 2.2(a), is the commercial successor to the i-HY hand, which was used in the DARPA Autonomous Robotic Manipulation Challenge [12]. It is an

[1]http://www.labs.righthandrobotics.com/reflex-hand-1

underactuated 3-finger hand designed to be simple, inexpensive and durable. The hand has 2 fingers and an opposable thumb, each of which are interchangeable and are comprised of a proximal and a distal segment. Under-actuation allows easier modelling in simulation and grasp planning [13], but if in-hand manipulation [14] is required, then a complex hand-shaped end-effector should be used. The 2 fingers are interconnected by gears and can swivel 0-$\pi/2$ degrees. or actuation purposes, it employs 4 Dynamixel servo motors, 3 of them are used to flex the fingers and thumb, while the fourth changes their pre-shape configuration. The fingers are interconnected by a set of gears and always move in sync with each other. Communication and control of the hand is achieved over an Ethernet connection via a ROS package provided by RightHand Robotics. The fingers and palm have built in "TakkTile" sensors, which can be used for grasp validation. The hand provides 3 pre-programmed grasps called "pre-shapes", which, coupled with high friction rubber coated fingers, allow it to securely grasp objects of various sizes and shapes. The 3 pre-shapes are: 1) *Parallel grasp:* fingers positioned at 0 degrees (parallel to each other and the thumb). When closed all 3 fingers interlace. It is useful for holding and operating heavy tools or other cylindrical objects. 2) *Spherical/Power grasp:* fingers and thumb at 120 degrees from each other. It is useful for grasping large objects or small objects which need to be manipulated in a circular pattern (e.g. screwing on a nut on a bolt) 3) *Pinch grasp:* fingers are spread as far as possible (π degrees). The thumb remains stationary while the fingers are closed and meet at the fingertips. This configuration does not provide a very strong grasp but allows for picking very small objects.

2.2.2 The Custom-made Unigripper End-Effector

The UniGripper end-effector was designed by the PRACSYS team in collaboration with Unigripper[2]. Its design was guided by the restrictions associated with handling objects inside shelves. A Kiva pod bin has about 20(width)x25(height)x40(depth) cm size. The two middle rows of bins have 20cm height. Given the depth, it is difficult to reach objects deep inside them without extending the robot's arms. So, the design of the end effector

[2]A Swedish company specialized in vacuum grippers for the packing and automation industry: http://www.unigripper.com.

(a) UniGripper design.

(b) Final Implementation.

Figure 2.3: Altering the UniGripper design to better fit the Amazon Robotic Challenge specifications

has a shaft of 13 cm that ends in the planar vacuum gripper in one extremity. The planar gripper is able to move up to 90 degrees according to an 1-DOF joint. The initial design was improved and redesigned by Unigripper to take the form depicted in Fig. 2.2(b).

The planar surface uses a Unigripper patented technology that allows grasping an object even when only part of the planar gripper touches it. For that, the planar gripper has many individual vacuum valves that, once an object is sucked by one or more of them, the others close, allowing the vacuum through just some of the valves to hold the object. A sponge glued to the planar vacuum gripper, with holes in front of each valve, allows grasping objects that do not expose planar surfaces. The design was implemented as shown in Fig. 2.3(a), but could only move to either one or the other extreme of the 90 degrees 1-DOF joint, not allowing for continuous changes. To increase the flexibility of the tool, the on-off vacuum actuator highlighted in Fig. 2.3(a) was switched to a Firgelli L16-R Miniature Linear Servo actuator shown in the final instantiation of Fig. 2.3(b). To control the gripper,

a hardware module based on the Micro Maestro 6-channel USB Servo Controller was built. Two channels were used to: (i) control the UniGripper's 1-DOF joint pose, and (ii) turn the vacuum on or off. A software module implements a ROS driver for the Unigripper gripper that uses the ROS actionlib for controlling the joint pose, and a ROS service to turn the vacuum on or off. Experiments revealed that the gripper was a bit too large and could not handle some of the heavier objects. It worked better when objects could be picked up from above as the presence of vacuum foam required the gripper to be pressed against the object.

The custom alternative to the Unigripper end-effector (Fig. 2.2(c)) uses most of the same hardware but replaces the planar surface with a smaller 3D printed air manifold with two suction cups. It is considerably smaller, which makes planning easier and can grasp light objects without toppling. The design features two suction cups since the redundancy helps with heavier objects.

2.2.3 The Unigripper Hybrid Gripper

The hybrid gripper (Fig. 2.2(d)) was designed in collaboration with Unigripper. It combines a long fingered parallel gripper and Unigripper's patented vacuum technology on the flat side of one of the finger. A vacuum suction cup was later added to the tip as well. The goal was to explore the benefits of having a gripper that combines several grippers into one. The hybrid is still being tested, however preliminary results suggest it is bulky and heavy. Due to the length of the fingers and the requirement for 0-14cm parallel finger range, the linear servo motors used are large and heavy. Despite the fact that the rest of the gripper is made of aluminium it is quite heavy. These shortcomings may outweigh the benefits of having a hybrid, however we believe that with lighter materials and improved design, a hybrid system will prove useful.

2.2.4 Custom High-flow Vacuum Gripper

Another in-house designed and built vacuum gripper (Fig. 2.2(e)), uses a low-vacuum, high-flow vacuum source (a powerful vacuum cleaner). It is constructed out of 3D printed plastic and laser-cut plywood materials for low weight and rapid prototyping. It has a long arm which allows it to reach deep into the pod and also a wrist 1-DOF, similar to the

Unigripper end-effector.

During the Amazon Robotics Challenge it became apparent that systems using a low-vacuum, high-flow vacuum source often outperform the high-vacuum, low-flow solutions common in industry. This became quite obvious in the cases when trying to pick porous and non-planar objects.

2.3 Vacuum Sources

Two types of vacuum are considered:

Air ejectors, driven by a 2 HP air compressor – High Vacuum (-75kPa) / Low Flow

This setup is one of the most commonly used ways to produce vacuum and highly popular in industry. Its biggest advantage is the ability to produce very large lifting, which enables grasping heavy objects, assuming a good seal can be made between the suction cup and object surface. The low flow requirements also allow for the use of small suction cups, which is important in space-constrained environments. The main shortcoming of the high-vacuum setup is its inability to deal with anything but minor vacuum leaks, which is the reason it just fails for porous, rough and non-planar surfaces. The hardware is also expensive and relatively complex, requiring an air compressor, air ejectors, air valves, etc..

Vacuum cleaner – Low Vacuum (-12kPa) / High Flow

The results from the Amazon Robotics Challenge proved that a low-vacuum high-airflow vacuum source, such as a vacuum cleaner was capable of producing results similar and in some cases better than the more commonly used industry, high-vacuum low-airflow solutions. The biggest advantage of the high-flow solutions is in their ability to pick up objects securely, despite large vacuum leaks which can be quite common in a warehouse picking setting. Vacuum leaks are common and sometimes impossible to avoid. They can be due to the nature of the object's material and it's inability to hold vacuum (e.g. cloth), the object's geometry making it impossible for the suction cup to form a good seal or simply imperfect

positioning of the suction cup (e.g. partial seal). In many cases, low-flow vacuum solutions fail as the leaks are too large and vacuum cannot be maintained. The hardware setup is also relatively inexpensive, requiring just a vacuum cleaner and a relay. The biggest limitation of high-flow vacuum is that the maximum lifting force produced is lower compared to low-flow vacuum and a larger suction cup is required.

2.4 Grasp Validation

Depending on the gripper, we approach grasp-validation differently. In the case of the ReFlex hand, we use the built-in, tactile and bend sensors to determine if an object has been picked up. For our vacuum grippers, additional sensors were needed. To validate grasps for grippers that use low-flow vacuum, we use vacuum sensors. Depending on the object, the type of surface it has, its weight and other factors different levels of vacuum ensure a robust grasp, so each object is tested and the minimum vacuum threshold is identified. In the case of high-flow vacuum, vacuum sensors do not work, since the changes in vacuum between successful and unsuccessful grasp are minimal. This is why we use a Mass Air-flow (MAF) sensor instead. The procedure for identifying the maximum airflow threshold is similar to the one for low-flow vacuum.

Chapter 3

System Integration

In this chapter we will describe in detail the system and all the modules we will try to evaluate.

3.1 Software Architecture

The idea behind the software architecture is to create a system that allows to easily replace any module without having to reconfigure the system again from scratch. For example, in order to evaluate the different end-effectors, it was necessary to create a software infrastructure that can easily adapt to changes in hardware.

- The lower layer contains necessary definitions and planning primitives such as the manipulator's kinematics, required geometries, obstacle definitions, roadmap queries, graph search algorithms and general end-effector definitions. This provides the required modularity that enables changes (i.e. different end-effectors, different motion planning techniques) without having to reimplement the entire pipeline.

- The middle layer consists of manipulation task planners that have the ability to combine planning primitives provided by the lower level into complete plans that satisfy a specific simple task like pick, place, move, transfer and release.

- The top layer consist of a planning application that can create queries and send them to the middle layer, combine the resulting plans and then propagate them to the manipulator.

The system is simulation driven, i.e., a simulation process is employed, which has the ability to request from the planning application to complete specific high-level tasks. The

communication between the simulation and planning applications is achieved through the Robotic Operating System (ROS). The layered structure described above allows to modify the behaviour of the system easily.

3.2 Simulation Application

Figure 3.1: Visualization of the simulation application.

A simulation application has the role of the coordinator and is responsible for communicating with the rest of the system components. Initially the manipulator, the Kiva shelving unit and the object models are loaded into the simulation. Precomputed roadmaps storing arm trajectories are de-serialized and the work order file is loaded. Internally a high level automaton is utilized to keep track of the current task as well as invoke the corresponding procedure when required. The states of the automaton correspond to high level concepts

like *move to initial position*, *move to sensing position*, *sense*, *pick and place* etc. Each of these states make the appropriate calls to the corresponding component. For example, the state *move to sensing position* will request from the associated planning application a plan that will move the manipulator from its current position to one of the pre-defined camera placements, while *sense* will send the collected RGB-D images to the perception pipeline, will wait for the pose estimation and finally will update the object's pose in simulation to agree with the estimated one. The visualization of the initial state of the simulation after loading all the required components is displayed in Fig. 3.1.

3.3 Task Planner

The task planner is the decision making part of the software. This means that the task planner is responsible to send the correct commands to the simulation application in order to complete each task as well as solve the high level problem we are attempting to approach as efficient as possible. In our scenario, the task planner is going to choose the order of the objects that are going to be removed from the Kiva pod or the tote. Choosing which object to attempt is not a trivial task as it is a dynamic problem since it is affected by the state of the real world (i.e. which objects are contained in every bin) and the end-effectors that are currently used (i.e. which end-effector is able to grasp the target object).

For example, consider the bin displayed in Fig. 3.2. Grasping the "crayola" is impossible since the "duct tape" has to be removed first. The task planner has to figure this out in real time during the execution and avoid wasting time trying to remove the "crayola" from the bin before the "duct tape" is removed. In Chapter 5 we will describe how we used the grasp planner in order to improve the overall performance of the system.

3.4 Pose Estimation

Pose estimation for objects inside the Kiva shelving unit is a challenging problem. Especially in a scenario that time is limited, a lightweight and fast perception solution is needed. Competing teams in the Amazon Robotics Challenge used software like the "Point Cloud Library (PCL)" [15], "Open Computer Vision Library (OpenCV)" [16], "Object Recognition

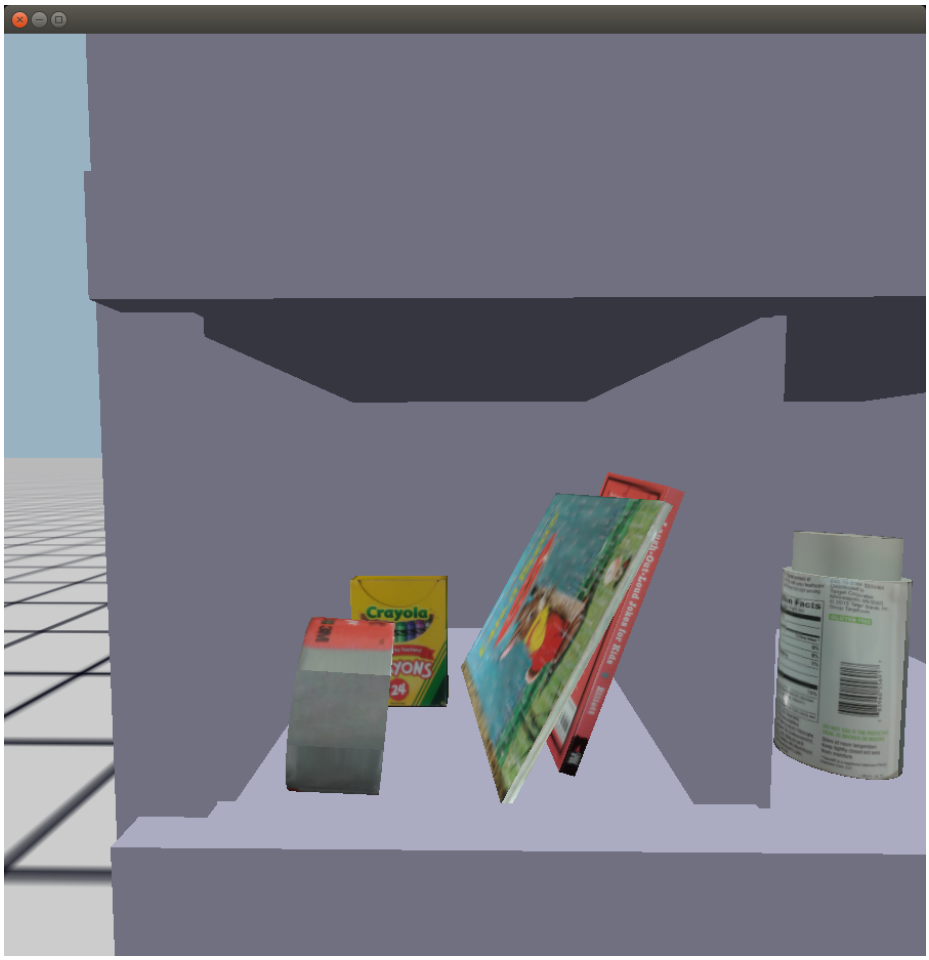


Figure 3.2: Objects in the back of the bin are blocked by the ones in the front.

Kitchen (ORK)" [17], "Simultaneous Detection and Estimation (SDS)" [18], "Linemod" [19], "Ecto"[20] and "Scikit Learn" [21].

To assist work in pose estimation, the Rutgers team generated a dataset including RGB-D data for all the ARC objects [22]. Originally, the solution considered corresponded to SimTrack [23], i.e., a pose estimation and tracking software that keeps track of the relative pose between the camera and the objects using predefined visual features. The longer the camera observes the object, the accuracy of the pose estimation increases. In order to get a good pose estimation for a target object, trajectories were recorded that moved the camera slowly in front of each bin. Because of the requirement to move the camera slowly in front of each bin in order to get an accurate estimation, this procedure was a bottleneck and instead a perception solution [24] using Regions with Convolutional Neural Network (R-CNN) was

developed. This faster approach uses 3 pre-defined camera placements in front of each bin allowing estimations even for partially occluded objects. An RGB-D image is captured at each camera placement and sent to the perception pipeline. After an initial pose estimation, physics are applied to the object in simulation in order to correct the pose estimated.

3.4.1 Brief Overview of the R-CNN Pose Estimation Software

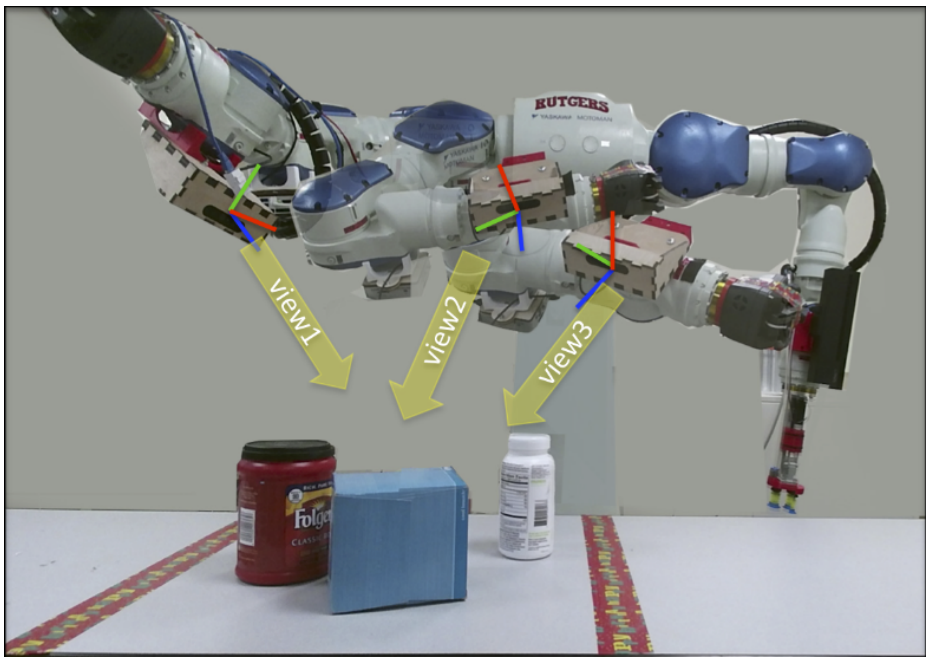


Figure 3.3: The 3 different camera placements used for generating learning data. [1]

Offline and before using the pose estimation software in a real world experiment 3D CAD models were generated for each of the objects we want to recognize and estimate their pose. A challenging procedure when using machine learning and convolutional networks is the training of the network. The initial approach was to initialize a physics aware simulator with the 3D CAD models to train the R-CNN. The scene was automatically altered and labelled in the simulator and the output data was used for training the network. When testing the R-CNN on real images, the success ratio for detection was high, although it failed for specific views and lighting. To compensate for these cases the decision to use real RGB-D images from real world scenes was made. This procedure is time consuming and

[1]Reprinted from "http://paul.rutgers.edu/ cm1074/"

requires a lot of manual labour. In order to do avoid having a person changing the scene manually and labelling it in order to creating a variety of scenes and training data, the manipulator itself was used.

A scene was set in front of the robot on a table-top. Then the manipulator moved the camera which was mounted on its arm in 3 pre-defined camera placements (Fig. 3.3) and captured an RGB-D image. Those images and point clouds were merged and labelled. The R-CNN trained with the simulated data was used in order to estimate the poses of the objects in the scene. A random object was chosen and the robot picked it and placed it in a different location and the above procedure was repeated (Fig. 3.4). That way labelled data were created and used for training the R-CNN. This data was used to train the R-CNN with real world data.

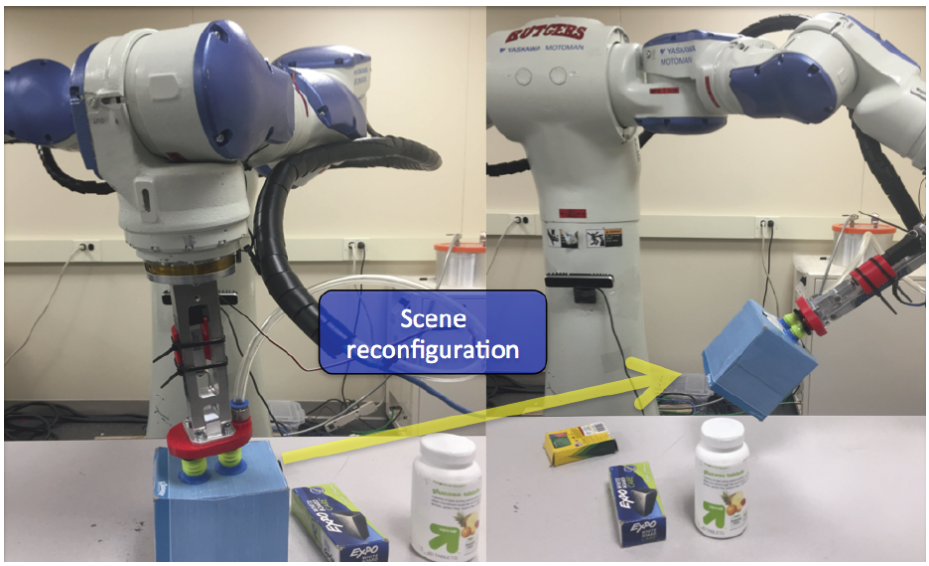


Figure 3.4: Reconfiguring the scene. [2]

During an experiment, the robot moves the camera in the 3 pre-defined camera placements and captures the point cloud for the whole scene. Then, the R-CNN is used to recognize the object we are interested in estimating its pose. The 3 point-clouds are merged and 3D segmentation is used to remove the resting surfaces and outliers from the resulting point cloud. Model matching registers the 3D CAD models to the segmented point cloud (Fig. 3.5) producing a 6D pose estimation for the object in question. Finally, to improve

[2]Reprinted from "http://paul.rutgers.edu/ cm1074/"

the estimation, after the pose of all the objects in the scene is estimated, physics are applied to correct possible errors in estimation. Testing and evaluating the pose estimation software is out of the scope of this thesis, so during the experimental evaluation we suppose that the poses of the objects in the scene is already known.

Figure 3.5: Matching the 3D CAD models to a real world scene. [3]

3.5 Grasp Planning

Many methods have been proposed, both on-line and off-line [25], [26], [27], to generate grasps given an end-effector and an object pose. A grasp is classified as valid if there exists a collision free IK-solution for the whole manipulator, such that the end-effector can be attached to the object's surface without colliding with the obstacle geometries around the object (i.e. Kiva shelving unit, other objects). After estimating the object's pose, the process is to:

- Create the set of *feasible* grasps in simulation by moving the end-effector's geometry, without the rest of the manipulator's arm, on the target object's graspable surfaces and rejecting the grasps with insufficient overlaps between the 2 geometries. The sampling process is guided towards the surfaces that are not hidden or away from the manipulator.

[3] Reprinted from "http://paul.rutgers.edu/ cm1074/"

- For each of the sampled grasps in the *feasible* set, "TRAC-IK" [28] is used to find collision-free IK solutions for the whole arm. If a solution is found that is collision-free, the grasp is marked as valid.

- For each of the valid grasps, motion planning is invoked to provide a plan that moves the end-effector to the grasping pose. If motion planning succeeds in providing a collision-free plan, the procedure is repeated for placing the object in a specific *target pose* (i.e. order bin).

The first step is to create the *feasible* set of grasps. We need to use a different approach depending on the end-effector. The reason for that is that a parallel end effector needs to be placed in a way that the fingers are surrounding the object before attempting to grasp, while in the case of a vacuum end-effector, in order to have a successful grasp, we need to bring the end effector on the surface of the object.

The initial approach was to generate offline grasping databases using "GraspIt!" [29] for each object. Because of the high volume of manual work required, the current system uses an on-line approach to get *valid* grasps that are dependent on the object pose during the execution of each experiment and invoke motion planning algorithms to move the end-effector to one of those valid grasps.

3.5.1 Generating a Database of Grasps for Parallel End-Effectors

The GraspIt! software [30], [31] was used to generate stable valid grasps for the "ReFlex" hand, which generates grasps by using a set of predefined "virtual contact points" on the 3D model of the end-effector (Fig. 3.6(a)). The planner samples various hand positions and orientations to bring the contact points close to the object surface. Depending on the contact point positions different types of grasps can be encouraged. Each grasp is then evaluated and scored according to a grasping metric [32], [33]. Such metrics consider the objects' 3D geometry and contact-point information, so as to detect high-quality grasps that can withstand disturbances [34], [35], [36].

Many grasps generated in this way are not feasible as they result in collisions between the end-effector and the environment. For example, the grasp of Fig. 3.6(b) will certainly

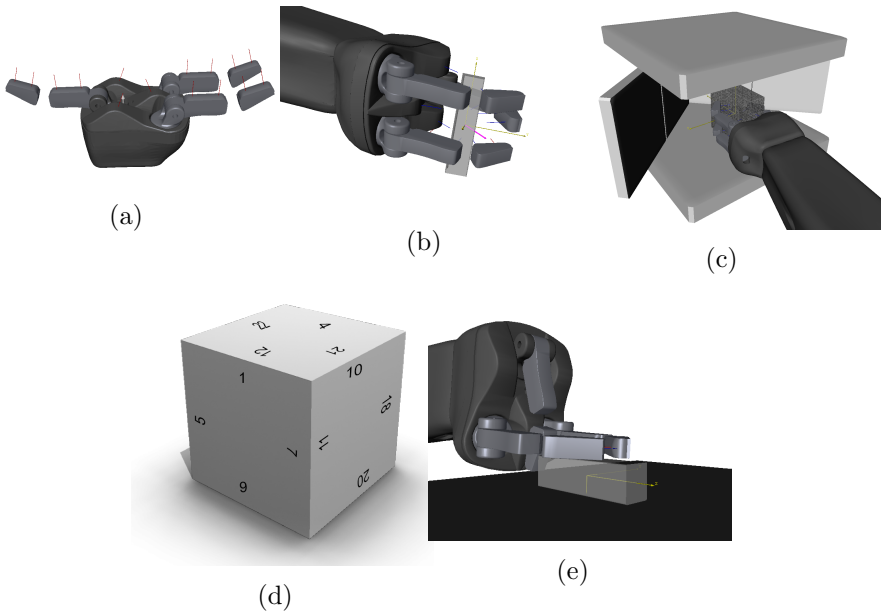


Figure 3.6: (a) The 'ReFlex Beta' hand in GraspIt! The red lines are visual markers showing the virtual contacts, corresponding to the friction cone, aligned with the contact normal. (b) Sample stable grasp generated without obstacles, with both the hand and object floating in free space. (c) Setup for the generation of the grasps in an environment including obstacles and the hand mounted on a virtual arm. (d) The 24 different axis-aligned object poses considered arise from the 6 sides of a cube that could face the hand and the 4 different orientations in each case. (e) A pinch grasp. This grasp would fail even though it is highly ranked by the eigengrasp Planner.

result in collisions with the resting surface unless the object is balancing on its small side, which is highly unlikely. To improve the quality of the grasping databases, such constraints were taken into account during the grasp generation process. A setup was used where the object is placed on a resting surface and the hand attached to a virtual arm link is only allowed to approach the object from a specific direction as in Fig. 3.6(c).

Then there are 24 different object poses for a cuboid that is axis-aligned that one can consider, as in Fig. 3.6(d). A different grasping database was generated offline for each such pose of an object inside the constrained setup of Fig. 3.6(c). Thirty grasps were generated for each pose with the exception of objects that it was not possible to reach this number. Online and once object pose recognition is performed, the 2 or 3 closest axis-aligned poses are identified and the corresponding grasping databases are loaded.

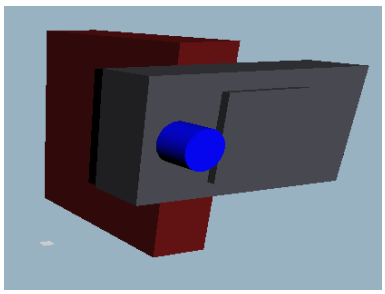
To simplify online planning, the ReFlex hand was limited to parallel grasps (Fig. 3.6(b)) and pinch grasps (Fig. 3.6(e)). Several different virtual contact definitions were used to

favour the generation of these types of grasps. The GraspIt! eigengrasp planner used the "Hand Contacts" and "Potential Quality" functions to rank grasps and executed 40,000 "Simulated Annealing" iterations. Upon completion, GraspIt! returned the top 50 pre-grasps ranked based on "grasp energy". Some of the resulting grasps, despite having a high score were not robust (see Fig. 3.6(e)) and they were visually pruned from the databases.

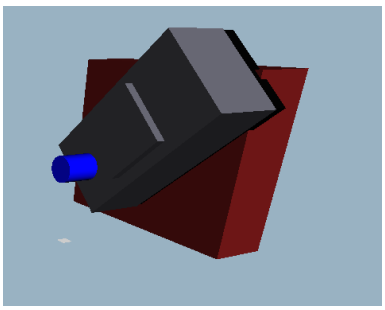
3.5.2 Generating a Database of Grasps for Vacuum End-Effectors

Similar to the Reflex hand, 24 Unigripper databases were generated for different axis-aligned object poses. Each object was approximated by a cuboid. As the vacuum solution requires part of the planar gripper surface to touch the object, the generation process simply samples grasps on each face of the approximated cuboid. For each pose, the object is considered to be inside a constrained setup similar to Fig. 3.6(c) and cannot be grasped from the bottom or the back. Grasps are sampled only for the front, top, left, and right face:

- Front face: 2 positions are sampled (p_0: face centre, and p_1: 1/6 of the face's height above p_0) and 8 orientations for each position (multiples of 45^o).
- Top face: 2 positions are sampled (p_0: face centre, and p_1: 1/6 of the face's length in front of p_0) and 3 orientations for each position (-45, 0, and 45^o rotations).
- Left/Right faces: 3 positions are sampled (p_0: face centre, p_1: 1/6 of the face's height above p_0, and p_2: 1/6 of the face's length in front of p_1) and 3 orientations for each position (-45, 0, and 45^o rotations).



(a) A grasp on the front face.



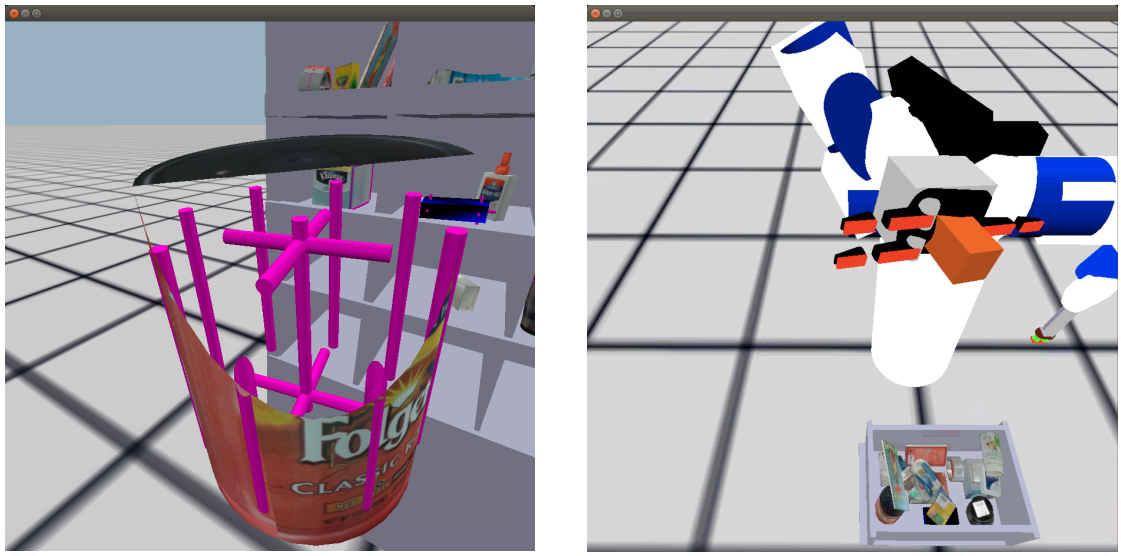
(b) A grasp on the left face.

Figure 3.7: Examples of Unigripper grasps in simulation

This results in 40 grasps per object pose. Grasps that collide with the resting surface or grasps that do not provide enough overlap between the gripper sponge and the object face are filtered out. The required overlap is set to be either half of the total area of the sponge or the area of the object face, whichever is smaller. After filtering, the resulting databases stored on average 30 grasps for each pose, equivalent to the ReFlex databases.

3.5.3 Online Generation of Grasps for Parallel End-Effectors

When working with parallel end-effectors we add cylindrical geometries around the mesh of each object (Fig. 3.8(a)) that we want to manipulate. We also create a cuboid that we place between the fingers of the end-effector. The cuboid is displayed in Fig. 3.8(b). Next we start sampling grasps so that the cuboid collides with the cylindrical geometries and at same time the fingers are not colliding with the object itself. In order to populate the *feasible* set with grasps that we can also bring the arm to, we limit the faces of the object depending on its pose. That means that we allow sampling only on the front, top and side faces of the object in respect to the manipulator's position.



(a) Grasp descriptors for parallel end-effectors. (b) Geometry defined between the fingers.

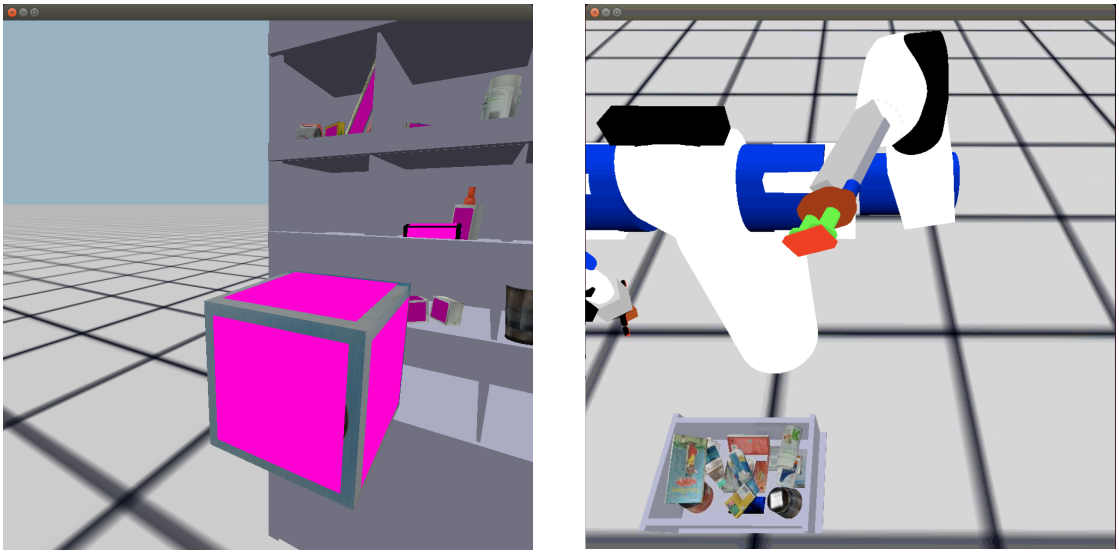
Figure 3.8: Generating grasps for parallel end-effectors.

The placement and size of that cuboid will affect the number and quality of grasps that will be generated, so coming up with the most optimal parameters can be a challenge. If the cuboid is placed close to the palm all the sampled grasps will have a higher probability that

will result in a successful grasp in a real world experiment, but at the same time planning for this kind of grasp will be more challenging as the end effector will need to go deeper in the Kiva pod. Also there is always a chance of tipping the object off with the end-effector resulting in a failed grasp.

3.5.4 Online Generation of Grasps for Vacuum End-Effectors

In order to generate grasps for a vacuum end-effector, we need to define a geometry on the vacuum surface of the end-effector. Notice the red rectangle in Fig. 3.9(b). Then, we also need to define geometries on the surface of the object as shown in Fig. 3.9(a) . Finally, a grasp is generated by moving the end-effector on the object so that the geometry on the vacuum surface overlaps with on of the geometries on the surface of the object. Similarly to the parallel end-effector case, we also limit the faces of the object that we are going to sample, so that we get grasps only on the front, top and sides of the object with respect to the manipulator.



(a) Grasp descriptors for vacuum end-effectors. (b) Geometry defined on the vacuum surface

Figure 3.9: Generating grasps for vacuum end-effectors.

A score can be used to choose the best grasps based on the overlap between the geometry on the vacuum surface and the geometries on the surface of the object.

3.6 Motion Planning

A variety of approaches for motion planning is available like "MoveIt!" [37], "OpenRave" [38], "Drake" [39], and "trajopt" [40]. For the experiments of this thesis an in-house built planning software [41], [42].

The goal is to come up with a motion plan that brings the end-effector to the current pose of the object, grasp it and then transfer it to the target pose. The final motion plan consists of 7 components:

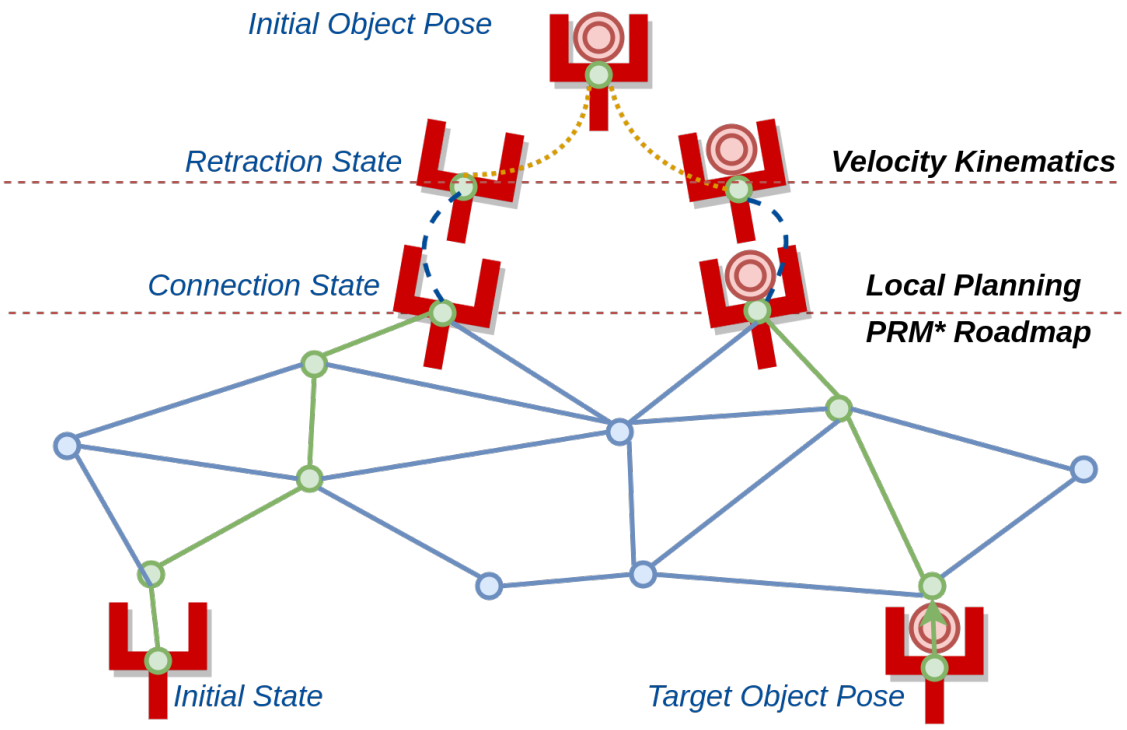


Figure 3.10: Motion Planning Pipeline

- **The Transit plan** brings the manipulator from the *initial* state to the *connection* state using a *PRM** roadmap, which was created off-line. The connection state is a state that belongs to the roadmap and at the same time is at close proximity to the target object.

- **The Transit Connection plan** brings the manipulator from the *connection* state to the *retraction* state. The retraction state is a state in the proximity of the target object but not on the roadmap. IK-based local planning is used to steer the end-effector to

the retraction state.

- **The Reaching plan** brings the manipulator from the *retraction* state to the surface of the object through Jacobian-based computation of velocity kinematics.

- **Secure grasp plan:** A real-world grasp might fail due to pose estimation errors. So, a *push* control is added to the direction the end-effector is facing. This plan is also computed by using Jacobian-based velocity kinematics.

- **The Retracting plan** brings the manipulator with the object to the *retraction* state. This plan is the reverse of the Reaching plan. This procedure brings the end-effector and the object away from the static or non-static geometries, simplifying the planning process from there to the object's target pose.

- **The Transfer Connection plan** attempts to bring the manipulator back to a node on the *PRM** roadmap. From there, it is possible to plan and bring the object to its target pose. To compute this plan, IK-based local planning and interpolation at the joint space is used.

- **The Transfer plan** brings the end-effector with the object to the target pose through the roadmap. The plan is lazily collision checked to ensure there are no collisions with other objects.

The same pipeline can be used for both the picking and the stowing tasks. The only thing that will change between those tasks is the initial and final position of each object. The motion planning pipeline is summarized in Fig. 3.10

3.7 Evaluating the System

Let's consider the modular system described above. A graphical representation of the components and modules of the system under evaluation is shown in Fig. 3.11. We observe that several modules are interconnected and it is not clear how the overall performance of the system will be affected if a module is changed or replaced.

A change on a module of the system can affect the performance of the whole system. However, even if the performance of a module is improved, it is not guaranteed that the performance of the whole system is also improved. Similarly, if the performance of a module decreases it may not affect the performance of the whole system. This is why having a way to evaluate a system and all of its modules as a whole is essential in order to achieve best performance as well as find out where are the bottlenecks and which modules should be further improved. This thesis proposes an evaluation platform that can be used to easily evaluate a complex modular system. The evaluation of a system can be both on the software used as well as the hardware.

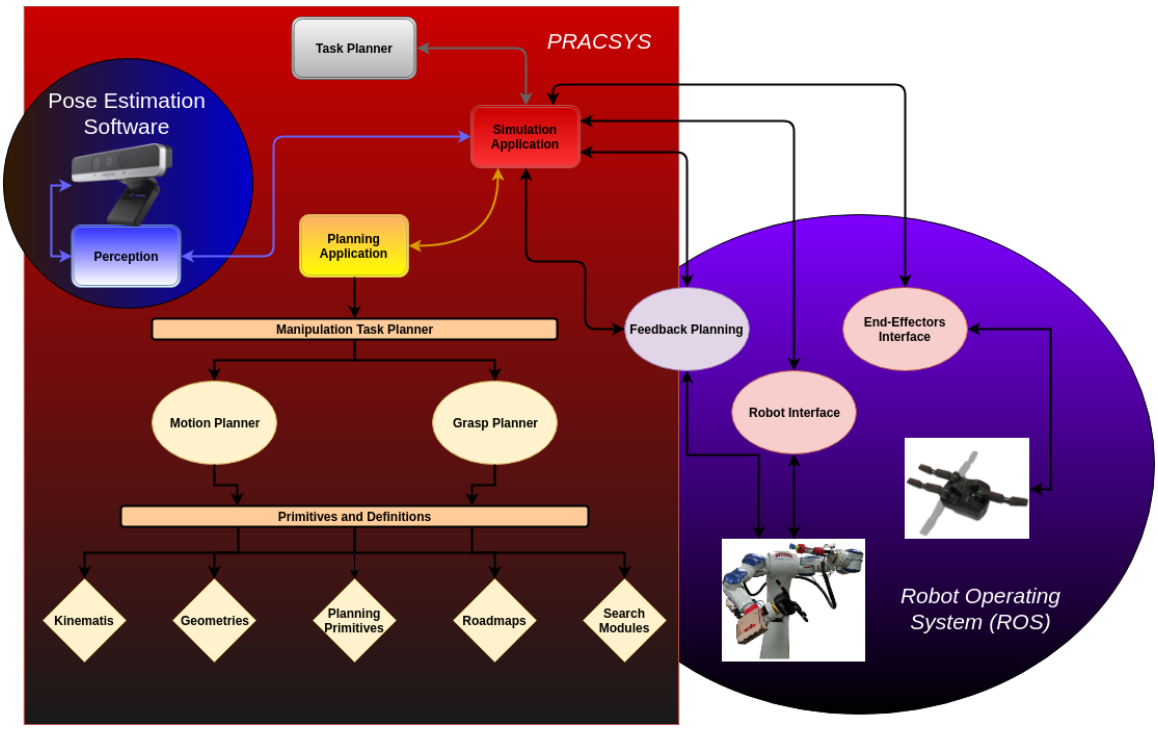


Figure 3.11: Scheme of the System Under Evaluation.

In order to evaluate the end-effectors we set up an experiment in an environment similar to the one of the Amazon Robotics Challenge but with only one object present in a pre-defined bin for each attempt. We tried to be as unbiased as possible regarding the different placements of the objects and not favour a specific end-effector of the ones we put under test. Along with the evaluation of the end-effectors we gathered statistics about the grasp and motion planning pipelines that showed us which modules caused the most failures. The

results of these experiments are presented in Chapter 4. Those results gave us indications of which modules needed refining or replacement which we took into consideration during our final approach described in Chapter 5.

Chapter 4

Hardware and Software Evaluation

In this chapter we will present the preliminary results of an experiment that was intended to help us make decisions on which of the available hardware performed better in a similar scenario to the Amazon's Robotics Challenge, as well as identify the bottlenecks in our software pipeline. In these experiments we use the Kiva pod and the Amazon objects, in each experiment only the target object is placed at the centre of a bin.

Taking into account the results of these experiments we modified our initial approach were it was necessary, improving the modules that affected the overall performance the most. The final approach is evaluated at Chapter 5 where the actual picking and stowing tasks of the Amazon Robotics Challenge are attempted.

4.1 Evaluating the End-Effectors

A grasping challenge is setup to evaluate the two end-effectors. Every experiment starts from a collision free state where the arms are outside the Kiva pod. The objects are placed in a specified pose inside a bin of the Kiva pod. The planning module loads the pre-computed grasps and tries to compute (i) a collision free trajectory for the target arm to a feasible grasping state for the current object, (ii) moving the object out of the bin, and (iii) dropping it at a target location outside the Kiva pod. The parameters that vary are the following:

- 2 end-effectors: *Unigripper* and *ReFlex* hand.
- 12 objects that vary in dimension, shape, weight, deformability, and material.
- 2 bins of the Kiva pod - the top middle bin and the one below it, which has a smaller height.

Table 4.1: Planning statistics

End Effector	Grasp success ratio	Planning success ratio	Solution quality (sec)	Comp. time upon success (sec)	Comp. time upon grasping failure (sec)	Comp. time upon planning failure (sec)	Average failure time (sec)
`UniGripper`	0.0449	0.6406	47.1128	3.3817	0.1014	12.5685	0.3098
`Reflex`	0.0333	0.6335	46.1100	3.3590	0.0993	12.4044	0.2552

Table 4.2: The planning success ratio for every object.

	spark plug	**cheezit**	**crayola**	**rolodex**	**elmers glue**	**feline**	
Unigripper	0.9487	0.1573	0.6327	0.6623	0.9565	0.5185	
Reflex	0.9667	0.2143	0.6552	0.8889	0.7879	0.5882	**Average**
	sticky notes	**kong duck**	**kyjen**	**mark twain**	**outlet plugs**	**duck toy**	0.6371
Unigripper	0.8214	0.8095	1.0000	0.4615	0.6667	0.7813	
Reflex	0.9412	0.6667	0.8750	0.3000	0.5789	0.9375	

- 3 poses in each bin starting with the pose shown in Fig. 4.1. The second pose is rotated 45° about $+Z$ (pointing up) and the third is rotated 90° about $+X$ (pointing to the Kiva pod) relative to the initial pose.

The offline generated grasping databases that were described in Section 3.5 were used in order to evaluate the 2 end-effectors. All these combinations are executed in simulation. For every object-end effector pair, there is a grasping database that corresponds to the set of relative configurations representing stable grasps. All the grasps are attempted. For the grasps, that can be reached with a collision free grasping state, the planning infrastructure tries to solve the motion planning problem to connect this state and solve the problem.

4.2 Planning Evaluation

All the grasps for the corresponding object/end-effector database are attempted. For successful collision-free grasps, the motion planner is invoked and the ratio of successful planning attempts is evaluated. The duration of the generated motions is used to measure the quality of the solution given similar maximum arm velocity. For every parameter combination, the two grasps that yield the best solution quality are evaluated in a real-world execution.

Table 4.1 summarizes the planning evaluation. The first column identifies the ratio of precomputed grasps, which are both reachable in terms of IK, collision free and provide

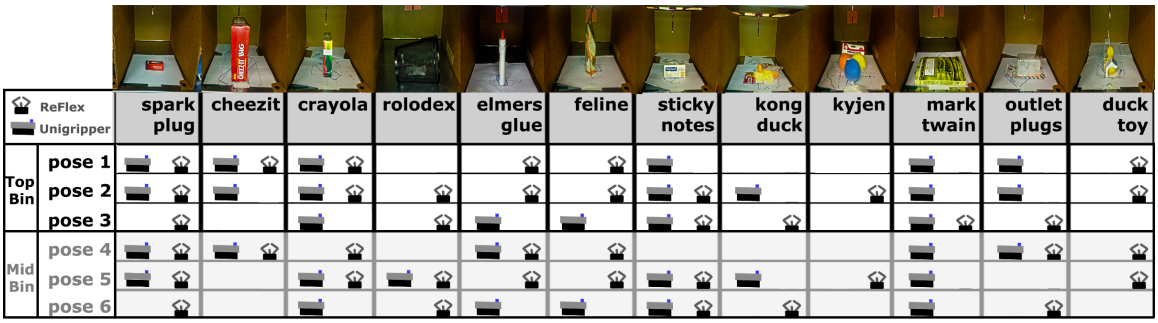


Figure 4.1: Successful grasps out of two attempts for each object/pose/bin combination. The objects are placed in pose 1 in the photos. If an object was successfully retrieved from the Kiva pod, the corresponding end effector image is present for that pose entry. (left: UniGripper, right: ReFlex)

collision-free reaching and retracting plans. About 4.5% of the precomputed grasps for the `UniGripper` and 3.33% of the grasps for the `ReFlex` end up being valid in this manner. Then, the motion planning phase solves the majority of the corresponding problems (63% to 64%), given a successful grasp. The resulting path quality is equivalent for the two end-effectors. Grasping evaluation is significantly faster than motion planning, both when it succeeds and fails. Planning takes 3 seconds on average when a plan is found and about 12 seconds upon a failure. But the average failure computation time is small given that most failures arise from the grasping evaluation. Computation times are similar between the two end effectors.

Table 4.2 shows that the problem is harder for bigger objects, such as the *cheezit*, *feline* and *mark twain*, which are difficult to maneuver out of the bin. The object pose also affects whether grasping succeeds. For objects, such as *mark twain*, *kong duck*, *elmers glue*, and *outlet plugs*, the *Reflex* hand will not have a grasp when the object has a face that is flush against the supporting floor and the fingers need to wrap around that face for a stable grasp. For objects like the *feline* and *elmers glue*, it is difficult to find grasps with the *UniGripper* in poses that have the narrow face facing outwards. Since certain faces are impossible to grasp with a suction surface, these narrow faces do not give rise to grasps in the databases.

4.3 Physical Evaluation

To setup the physical evaluation, the poses used in planning were reproduced in the real-world. The Amazon-Kiva pod was placed at the specified distance in front of the robot

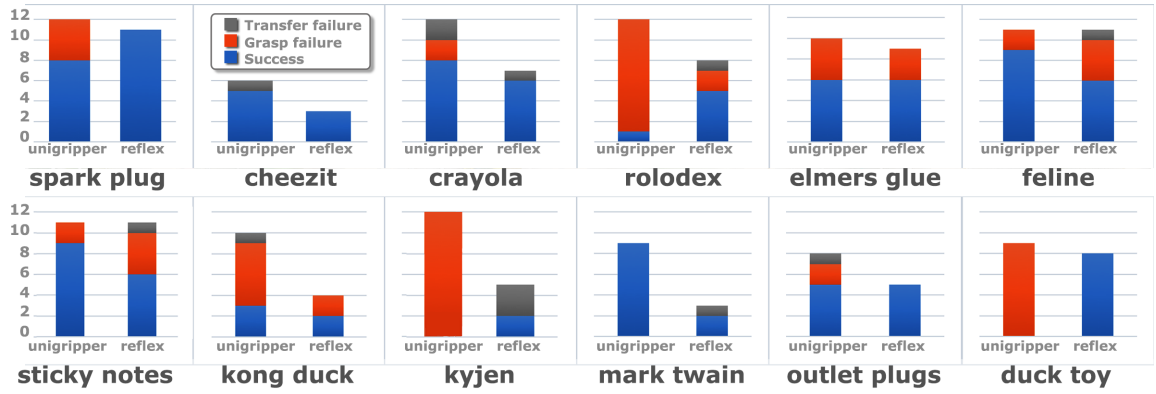


Figure 4.2: A visual breakdown of the performance of the two end-effectors in real-world experiments. The number of successful grasps and retrievals from the Kiva pod is plotted, along with the number of failures with respect to grasping the object and transferring it.

and each objects was placed inside the bin at the corresponding pose. The experiments consisted of replaying the two best quality solutions for each pose provided by simulation, and logging whether a grasp was successful in the real world. A grasp may fail either because the object was moved by the end effector before grasping, or because the object's shape or target grasp area prevented a successful grasp. Furthermore, even if the object was grasped it may be dropped during the transfer from the bin to the target location. In Figure 4.2, one can observe the success count for each end effector and object tested as well as the reason a certain solution failed.

Figure 4.1 demonstrates that, depending on the surface properties and morphology of each object, the use of a specific end effector was more advantageous than the other. For instance, the "mark twain" object, was successfully grasped by the Unigripper for all executions attempted, since it is physically easier for a vacuum gripper to grasp an object with flat surfaces. The ReFlex hand, however, is unable to grab a book while lying on a flat surface. On the other hand, "rolodex" was almost impossible for the Unigripper to grasp given its mesh surface, while the ReFlex was successful most of the times. Even for cases that solutions were provided from planning for every pose of an object (i.e., "kyjen"), Unigripper was unable to perform a successful grasp as the object does not expose a large enough flat surface.

Overall, 122 trajectories were tested for the UniGripper in real-world experiments [1].

[1]Videos can be found in: https://drive.google.com/open?id=0Byh0TCcqXnO-ZzF6QWdlVVJxVm8

Out of those, 63 succeeded. The majority of failures arise from three objects ("rolodex", "kyjen" and "duck toy") for which it is difficult to establish suction. It is easy to include this knowledge during the planning process and not generate such solutions. For the ReFlex hand, 85 trajectories were computed, which is a smaller number, showing it is harder to plan for this end-effector. Out of those, however, 62 trajectories were successful. These failures are more equally distributed over objects and more unpredictable. Out of the 78 considered poses (2 bins x 3 poses per bin x 12 objects), it was possible to find a successful grasp with either end effector 57 times.

Chapter 5

Evaluating the Final Approach

After evaluating the findings of the experiment described in Chapter 4, we noticed that the biggest bottleneck was the grasp planning approach. By creating grasping databases as part of our pre-processing procedure, we were very dependent on the quality of the each recorded grasp. Furthermore, only a small percentage of grasps, if any, were valid for the target object and if motion planning failed for these, there was no way to recover. For that reason we swapped our grasping database approach for an online grasping generation one as described in Chapter 3. This change though had an impact in the computation time spent upon planning failure, since we could keep generating new grasps and try to come up with a motion plan in difficult instances of narrow passage problems. An obvious but not efficient solution was to set an upper time limit for each attempt on a target object, but a more effective solution had to be found. Last but not least, since we introduced more objects in every bin resulting in partially or totally occluded objects, an efficient task planner that would choose which object was available for grasping and which objects were blocked had to be implemented.

The experiment consists of 2 tasks:

- The picking task where the objective is to remove all the objects from the Kiva pod. As described above multiple objects will be placed in the same bin, increasing the difficulty of the task.
- The stowing task where the objective is to remove all the objects from a densely packed tote in front of the robot.

The time available to complete each of the tasks is 15 minutes (900 seconds).

In each of the following sections a different approach in solving the Amazon Robotics Challenge is described and evaluated. For each approach we will present:

- A table containing the average failure time, the average success, the average quality of the solutions in seconds, the average planning time, how many objects were picked on the first try and the total time spent on the task.
- A chart showing the number of feasible and valid grasps for each object
- A chart showing the timestamps each objects was picked up and removed from the pod/tote
- A chart showing how the total time was allocated between the different objects.

The elements that will vary between each of the experiments are:

- The size of the roadmap used and
- The task planner intelligence

We will use the results of each approach to improve our final implementation which will be described at the end of this chapter.

5.1 Roadmap Size Effects

The size of the roadmap used affects both the efficiency of the final approach and the quality of the motion plans. As shown from the preliminary results described in 4, a significant amount of time is spent on motion planning failures. The motion planning pipeline tries to find a collision free path passing from the nodes of the roadmap using A*. The size of the roadmap affects the amount of time needed for A* to traverse the roadmap. On the other hand a small roadmap affects both the quality of solutions and the overall success rate of the system. We will try to measure the effects of the roadmap in the Amazon Robotics Challenge scenario by using a small 100 node roadmap, and a large 30.000 node roadmap. Since we identified the vacuum end-effector as the most efficient in our preliminary experiments, we are going to perform this round of experiments using only that. The initial placement of the objects in the Kiva pod is shown in Fig. 5.1

Figure 5.1: Initial placement of the objects in the Kiva pod for the picking task.

5.2 Task Planner Modes

The task planner we use can run with 3 different modes. Each new mode was developed to improve on the previous and attempt to increase the quality and the efficiency of the final approach.

Round Robin Mode

Our initial and simplest approach is that the task planner will choose which object to attempt based on a random order. A list is compiled containing all the objects in the shelf. The task planner will start going through that list and choose the target object based on its position in the list. After the target object is provided to the simulation application from

the task planner, the planning pipeline will be allowed to try to find valid grasps and motion plans for the pre-specified amount of time of 20 seconds. During that time the simulation application will:

- Invoke the grasp planner to come up with a set of valid grasps. The grasp planner will sample a pre-specified number of grasps and will try to find IK solutions for these. If an IK-solution is not found, the grasp planner will be invoked again to sample more grasps.
- If at least one valid grasp is found, motion planning is invoked and tries to come up with a motion that will guide the end-effector to that grasp.
- If the motion planner succeeds, the attempt is classified as successful and the simulation application executes the plan and the statistics for that attempt are sent to the task planner.
- If the motion planner fails, the statistics are sent to the task planner and the simulation application waits for the next target object.

When that time runs out, or a solution is found, the next in line object is chosen as a target. Every time a success occurs the target object is removed from the list. The task ends if all the objects are successfully removed or the 15 minutes time runs out.

Smart Mode

After gathering statistics with the task planner running in Round Robin mode, we observed that a significant amount of time, specifically 20 seconds, was spent trying to come up with grasps for objects that were impossible for the end-effector to reach because they were blocked by other objects. To avoid that we implemented logic that allowed the task planner to mark objects as blocked. If the grasp planner did not find any valid grasps for a target objects, the object was marked as blocked and the next target object was chosen. In order to mark the blocked objects as non blocked we used the information of which objects are in each bin. When a successful pick occurred from a bin, all the blocked objects *in that bin*

were marked as unblocked. As a result the grasp planner spent the full 20 seconds on an impossible to grasp object only once until that object was potentially unblocked.

Smart Fast Cycle Mode

Our third and final iteration tackled the fact that we were spending 20 seconds to identify a blocked object. Considering that an object could not become unblocked even if another object was removed from its bin (i.e. the object was blocked by multiple objects), we were still spending too much time attempting to grasp blocked objects. To avoid that, our third iteration of the task planner allowed the grasp planner to sample feasible grasps and attempt to find an IK solution to these only once. This means that for each attempt on a blocked object it was impossible to spend all 20 seconds trying to sample valid grasps and indeed the average failure time was significantly reduced.

5.3 Planning Without Obstacles

Initially we will execute the picking task without the Kiva pod. The objects are placed in front of the robot as if they were in the bins of the Kiva pod, but the actual Kiva pod is not present in the experiment. That way we can make sure that the online grasp planner can generate grasps for all the objects and the motion planner can generate plans that will solve the problem without the introduction of narrow passages (i.e. the bins of the Kiva pod).

Roadmap Size: 100/30.000, Task Planner Mode: Round Robin, No Kiva Pod

The overall statistics of the experiment runs without the Kiva Pod are presented in Table 5.1. The grasp planner found valid grasps for all the objects except for the "rolodex" object. This is expected because of the placement of that object as shown in Fig. 5.1. Although by a small margin, we notice that the overall performance of the small roadmap is better that the large one. This is expected since the problem without the introduction of the Kiva Pod is not challenging for motion planning so the small size of the roadmap increases the traversing speed of the A* algorithm and results in faster solutions. Note here that since

the "rolodex" object could not be picked by the vacuum end-effector so the experiment was *manually* stopped after the rest of the objects were removed from the Kiva Pod.

Table 5.1: Planning statistics - Picking Task - No Kiva Pod

Task Planner Mode	Roadmap Size	Total Picks	First/Last Pick (sec)	Avg Failure Time (sec)	Avg Success Time (sec)	Avg Planning Time (sec)	Avg Solution Quality (sec)	Total Time (sec)
RR	100	14/15	**5/174**	20.094	**2.292**	**6.248**	**5.892**	**194.981**
RR	30.000	14/15	7/192	**20.089**	2.780	6.627	6.698	213.070

5.4 Introducing Obstacles

5.4.1 Picking Task

We will now introduce the Kiva Pod to our experiment. Table 5.2 shows the overall performance of the system during the experiments with the 3 modes of the Task Planned and the 2 different sizes of roadmaps.

Table 5.2: Planning statistics - Picking Task

Task Planner Mode	Roadmap Size	Total Picks	First/Last Pick (sec)	Avg Failure Time (sec)	Avg Success Time (sec)	Avg Planning Time (sec)	Avg Solution Quality (sec)	Total Time (sec)
RR	100	11/15	43/782	**10.783**	**2.417**	**9.588**	14.476	897.518
RR	30.000	**12/15**	**39/607**	18.014	3.228	14.602	**10.613**	886.668
Smart	100	12/15	59/783	**3.690**	**2.238**	**3.603**	12.913	891.907
Smart	30.000	**14/15**	**39/574**	14.460	4.102	10.644	**12.149**	**574.534**
Smart-FC	100	12/15	**30/325**	**2.645**	**2.406**	**2.633**	16.388	885.550
Smart-FC	30.000	**14/15**	38/468	11.350	5.462	8.852	**12.629**	**468.925**

Overall Evaluation

From the planning statistics presented in Table 5.2 we notice that the only cases we removed all possible objects (all but the "rolodex" object) from the Kiva Pod was when we used the

smart and smart-FC modes of the task planner along with the large 30.000 node roadmap. The smart task planner solved the task in 574 seconds while the smart-FC task planner required only 469 seconds. We also notice the big average failure time when the round robin mode is used, and confirm that the implementation of the 2 other modes of the task planner was necessary to solve the task.

The quality of solutions was generally better when the large roadmap was used which is expected since using a small roadmap leaves a big part of the motion planning to the local planner which can come up with unnecessary motions in the workspace of the manipulator. The difference between the average solution quality when using the small roadmap in the cases of the smart and smart-FC task planner modes, can also be explained by the inconsistency of the local planner.

On the other hand, the failure, success and average planning times were significantly smaller in the cases we used the small roadmap. Specifically in the failure cases, the search in the small roadmap takes a lot less time compared to the case we have to search 30.000 nodes to declare failure. Failing fast for an object allows us to try more objects in the same amount of time, giving us the opportunity to remove an object that is easier to pick and making the overall task easier to solve. It is worth going into detail and see what happened in the other runs as we will discover some interesting findings In the following sections we will try to interpret the statistics we gathered from the combination of the different modes and different sizes of the roadmap.

Grasp Planning Evaluation

As shown in Figures A.7, A.8, A.13, A.14, A.19, A.20 the grasp planner comes up with significantly less grasps compared to the experiments run without the Kiva Pod (Figures A.1, A.2,). This happens because the majority of the sampled grasps are invalidated due to collisions with the Kiva Pod and/or other objects. The number of feasible/valid grasps shown in the aforementioned figures refer to the last attempt of each object during the experiment. We also notice that once again we get no valid grasps for the “rolodex” object. Generally the performance of the grasp planner is consistent between the experiments with some fluctuation depending on the order of the objects that were removed from the Kiva

Pod.

Motion Planning Evaluation

The first thing we notice is the large average failure times in the case of the round robin mode of the task planner. As mentioned in Section 5.2, when using the round robin mode we allow a specific time to the grasp and the motion planner for each attempt. Even if the object is blocked and no valid grasps are returned, the planning pipeline will keep trying to come up with valid grasps. Moving to the the smart mode we notice that the average failure time is significantly reduced by 65% in the case of the small roadmap and 20% in the case of the large roadmap. That reduction in the average failure time is due to the fact that we no longer try to plan for blocked objects since we can identify and skip them until another object is removed from the same bin. This means that in the case of failure most of the time was spent trying to come up with a valid motion plan for a valid grasp rather than trying to find a valid grasp for a blocked object. We further improve the average failure time by an additional 30% and 27% for the small and large roadmap respectively when using the smart-FC mode of the task planner. In both cases we also notice an improvement in the average planning time which is natural since the average planning time metric is affected directly by the average failure time.

Focusing on the first picked object timestamp of Table 5.2 we notice that although the timestamps when using the large roadmaps are consistent, that is not the case when using the small ones. That can be explained from the fact that the local planner that is used more in the case of the small roadmap is more susceptible to fail and additionally the quality of the resulting motion plan can be significantly worse. An interesting fact when looking at the last picked object timestamps in the case of the smart-FC task planner is that when using the small roadmap the last successful pick took place 325 seconds after the beginning of the experiment. The remaining 575 seconds were used to try and plan for an object that was difficult for the local planner to solve. Fact that can be validated when looking at Figure A.23. The 2 remaining objects in the Kiva Pod are the “I am a bunny book” and the “laugh out loud joke book”. The task planner correctly marked the latter as blocked and failed 216 times to pick the “I am a bunny book” from bin for a total of 562 seconds. This

is an indication that the overall performance can be improved by combining 2 roadmaps of different size in a single experiment. That way we can take advantage of the fast planning times of a small roadmap and pick as many objects we can as fast as possible and when we start to get many failures for a specific object, load the large roadmap and try to pick the more challenging objects. Comparing the Figures A.23 and A.24 we notice that until the 12^{th} pick the small roadmap outperforms the large one. Similar are the observations when comparing the experiments ran with the smart mode of the task planner (Figures A.17 and A.18).

We note the importance of reducing the planning time as much as possible, since we want to allow the Pose Estimation software as much time as possible in order to get an as accurate as possible estimation of the objects.

5.4.2 Stowing Task

Moving to the stowing task, we present the results of the experiments in Table 5.3. Fig. 5.2 shows the placement of the objects in the tote. Notice that because of the placement

Figure 5.2: Initial placement of the objects in the tote for the stowing task.

of the "rolodex" we now expect to be able to grasp it with the vacuum end-effector used. The experiments performed for the picking task showed that the best performing mode of the task planner is the smart-FC which is the one used during the stowing experiment.

Table 5.3: Planning statistics - Stowing Task

Task Planner Mode	Roadmap Size	Total Picks	First/Last Pick (sec)	Avg Failure Time (sec)	Avg Success Time (sec)	Avg Plan-ning Time (sec)	Avg Solution Quality (sec)	Total Time (sec)
Smart-FC	100	15/15	18/298	**2.672**	**2.734**	**2.691**	10.918	298.346
Smart-FC	30.000	15/15	**14/258**	3.153	3.977	3.528	**9.453**	**258.225**

Overall Evaluation

The task was completed when using both the small and the large roadmap. Overall, both of the roadmaps performed similarly as shown in Figures A.27, A.28, A.29 and A.30. Similarly to the picking task, the average failure success and overall planning times reported when using the small are better than the ones reported when using the large roadmap but only slightly. The overall time, although, is 40 seconds faster in the case of the large roadmap. This can be explained from the fact that the small roadmap failed more times when motion planning as well as from the fact that the average quality of the motion plans was worse compared to the large roadmap's.

Grasp Planning Evaluation

Comparing the Figures A.25 and A.26 we can see that the grasp planner is consistent since the number of valid grasps for each object in both cases are similar.

Motion Planning Evaluation

When using the small roadmap, the average failure, success and planning times are consistent with the picking task. That is not the case though when using the large roadmap. We notice that the reported times are 60% better compared to the times of the picking task. That can be explained if we consider that removing an object from the tote is an easier planning problem compared to removing an object from a bin since the bin is significantly smaller. Although, since our experiments included only removing objects from the tote and not placing them inside the Kiva Pod, we expect that when the complete stowing task is attempted the times will be closer to the ones reported during the picking task.

Chapter 6

Conclusion and Future Improvements

6.1 Re-Evaluate the System with Online Pose Estimation

The results presented in the previous chapters use the assumption that the objects' pose is already known. In the real Amazon Robotics Challenge though, each team had to estimate the pose of each object in real time with a pose estimation. The estimation of the pose is a procedure that takes time and the actual estimation is susceptible to noise. As a result, even if the planning pipeline comes up with a collision free plan to the estimated pose, the end-effector might fail to grasp it. To recover from failed grasps like that, online grasp validation might be employed as well as visual servoing with a small camera mounted on the end-effector to assist in grasping. It is worth mentioning a real world problem we faced during the execution of some preliminary experiments including pose estimation. There were cases that the end-effector was almost touching the object but could not grasp it due to small errors in pose estimation. For that reason we introduced the *push control* described in Section 3.6 that will move the end-effector 2cm towards the direction of the object in order to compensate for that error.

6.2 Introduce Rearrangement

In some cases the target object might be inaccessible due to other objects. In that case, the task planner with the assistance of the pose estimation software should include logic that will command the planning pipeline to remove the occluding objects and place them in different bin of the shelf in order to access the target object. The implementation to this is not trivial and is very dependent in pose estimation. In order to address this issue many questions have to be answered like, where is the best position to place the occluding objects

so that no more target objects are occluded. Implementing rearrangement logic adds an additional layer of complexity to the system since in order to grasp a target object more than one collision free plan must be found be the planning pipeline.

6.3 Discovering Available Bin Space

The stowing task results presented in the previous chapters only included the part of removing all the objects from the tote. The next step in order to complete the task is actually place them in the Kiva pod. Completing this task can be a challenge, especially if the Kiva pod already contains other objects. In cooperation with the pose estimation software a logic must be implemented to discover available space in the bins in order to place the objects from the tote. The same logic can also be used with the rearrangement procedure discussed in the previous section. It is worth mentioning that during the Amazon Robotics Challenge, placements of objects inside bins with more items awarded more points. This suggests that feedback planning might provide better as opposed to collision free motion planning. This means that the manipulator will try to push the target object slowly inside a bin with objects without checking for collision free paths as long as the feedback sensors report resistance forces below certain thresholds.

6.4 Defining Object Preference

By analysing the experimental evaluation of the previous chapters, we notice that the success ratio of the planning pipeline was higher for specific objects. This information should be included and a high level logic should be implemented that will give priority to those objects. Removing objects from the Kiva pod makes the problem easier and less constraint for the objects remaining. Furthermore, the rules of the Amazon Robotics Challenge, in case of a draw between 2 teams, give the advantage to the team that removed their first target object faster.

6.5 Switching Between Different Roadmaps

The proposed system at the time of evaluation did not have the ability to load more than one roadmaps. The roadmap that was chosen in the beginning of the experiment was the one that was going to be used during the experiment. From the experimental evaluation of the system we observe that roadmaps with different size perform better on different circumstances. For example it would beneficial time-wise to initially use a small roadmap since the time required to come up with a collision free solution as well as the average time required to fail is considerably lower compared to the one of a big roadmap. That way all the objects that are not a challenge to plan to will be removed from the scene, leaving the most challenging to attempt with a bigger roadmap.

6.6 Future Gripper Designs

Choosing the correct end-effector for the task is one of the most important choices that have to be made in every manipulation problem. The software and the planning pipeline will greatly benefit from an end-effector that is easy to plan for. We showed that for narrow passage problems like the Amazon Robotics Challenge, a vacuum end-effector attached to a long shaft has the best performance compared to a bulky hand shaped end-effector. Although there are object with specific texture, like the "rolodex" that a vacuum end effectors is almost impossible to grasp.

We are currently working on a new version of Unigripper solution with suction cups (Fig. 2.2(c)). We are experimenting with 10mm diameter suction cups (versus the old 35mm cups) and the results are promising. While lifting force is reduced it still seems sufficient for the heavier objects in our set. The smaller cup size will result in less vacuum leaks and will allow for much smaller overall size, making planning easier. We are also working on several soft-robotics inspired grippers [44], which will be able to provide robust grasps and handle collisions without getting damaged or damaging the objects or the environment. This will allow us to run learning algorithms with minimal supervision.

Appendix A

Detailed Statistics

A.1 Without the Kiva Pod

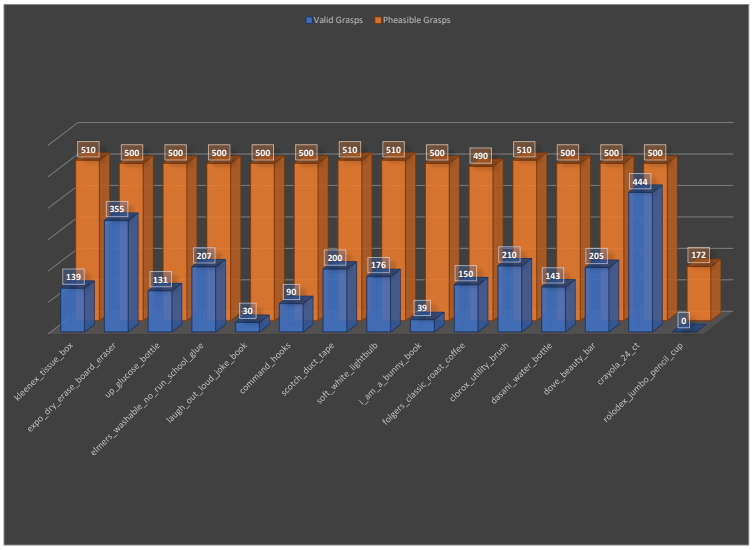


Figure A.1: Feasible vs Valid Grasps. Roadmap Size: 100, Task Planner Mode: RR

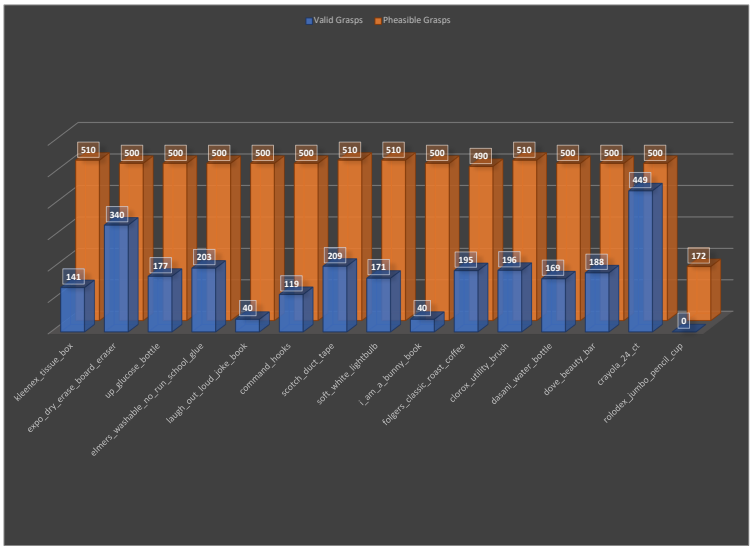


Figure A.2: Feasible vs Valid Grasps. Roadmap Size: 30.000, Task Planner Mode: RR

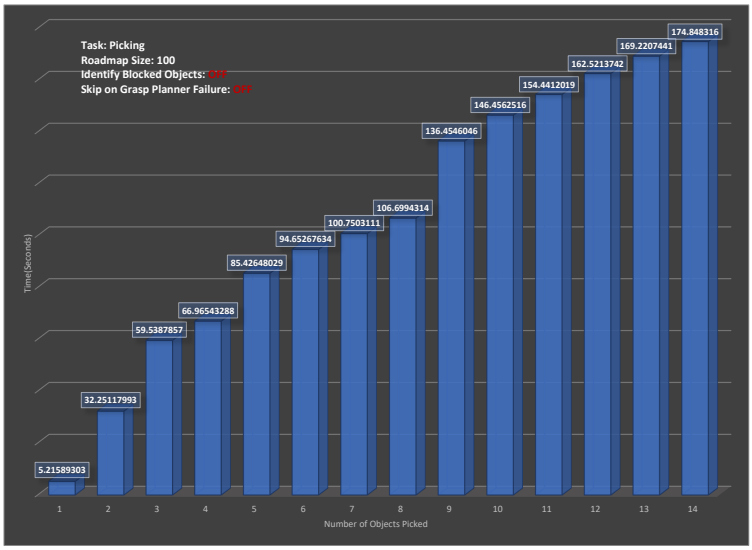


Figure A.3: Timestamp for each Pick. Roadmap Size: 100, Task Planner Mode: RR

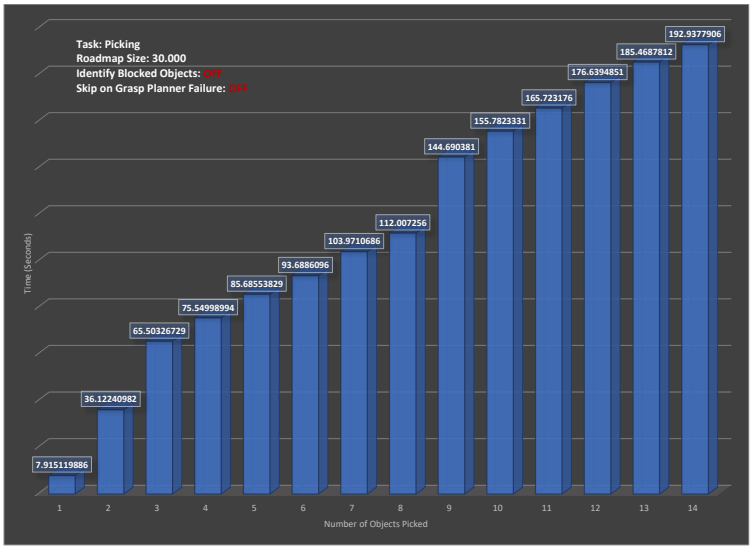


Figure A.4: Timestamp for each Pick. Roadmap Size: 30.000, Task Planner Mode: RR

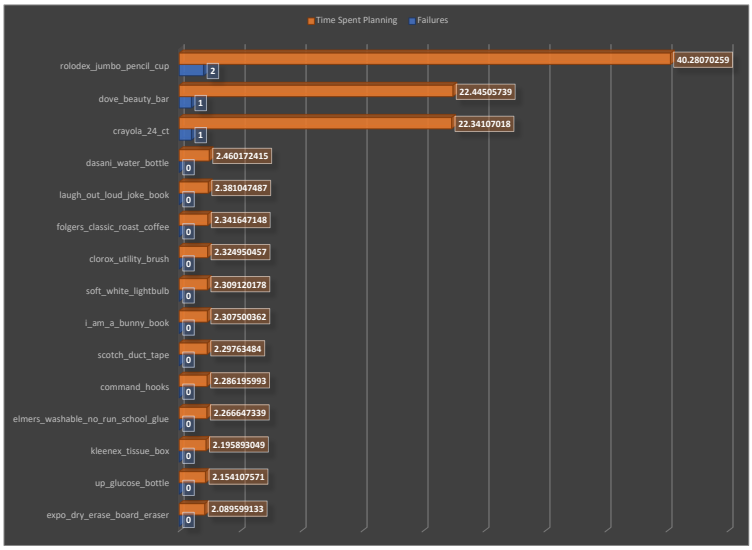


Figure A.5: Time Spent per Object. Roadmap Size: 100, Task Planner Mode: RR

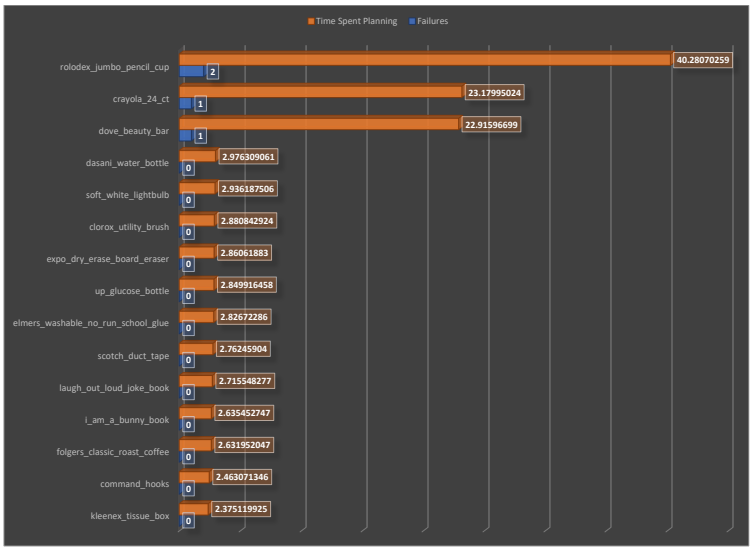


Figure A.6: Time Spent per Object. Roadmap Size: 30.000, Task Planner Mode: RR

A.2 With the Kiva Pod

Task: Picking, Task Planner Mode: Round Robin

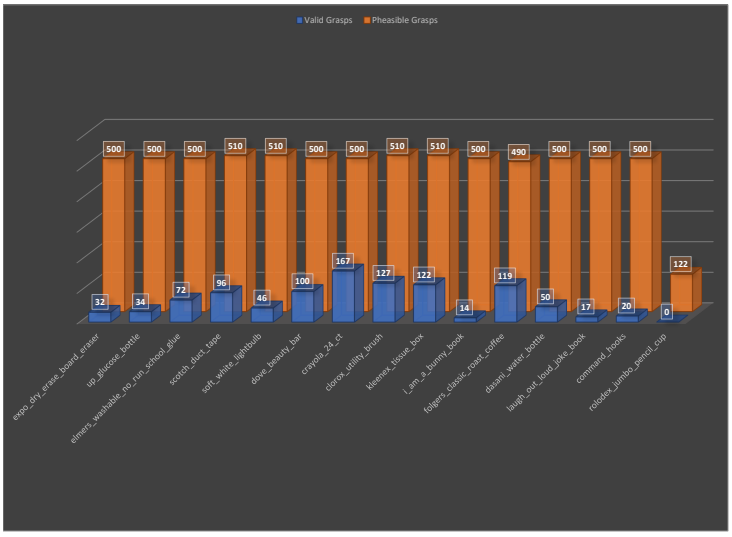


Figure A.7: Feasible vs Valid Grasps. Roadmap Size: 100, Task Planner Mode: RR

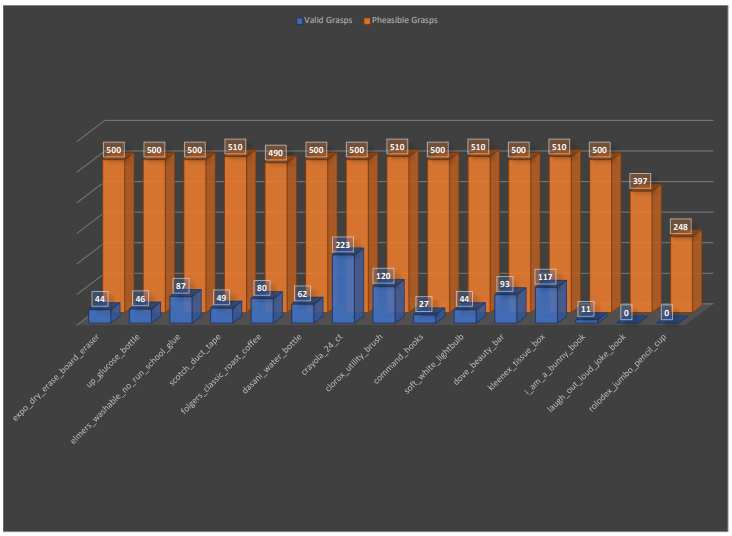


Figure A.8: Feasible vs Valid Grasps. Roadmap Size: 30.000, Task Planner Mode: RR

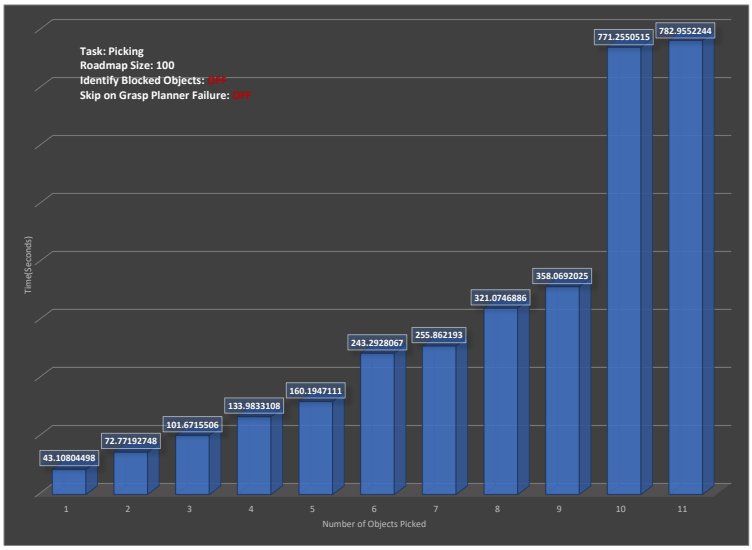


Figure A.9: Timestamp for each Pick. Roadmap Size: 100, Task Planner Mode: RR

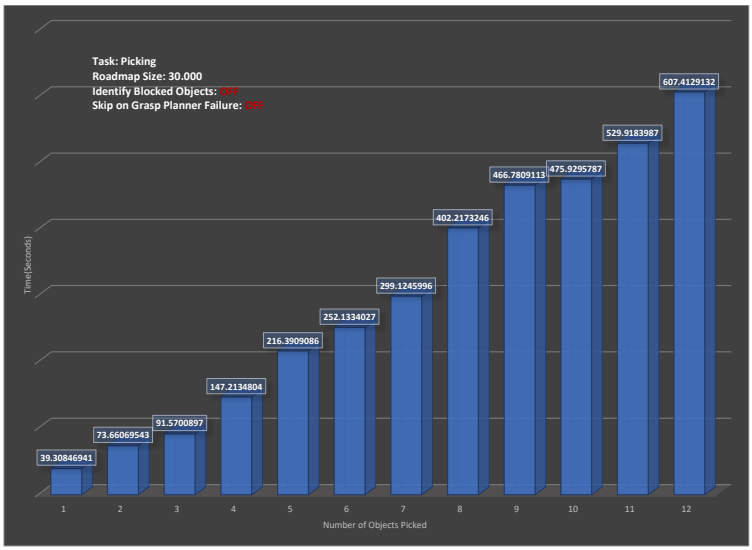


Figure A.10: Timestamp for each Pick. Roadmap Size: 30.000, Task Planner Mode: RR

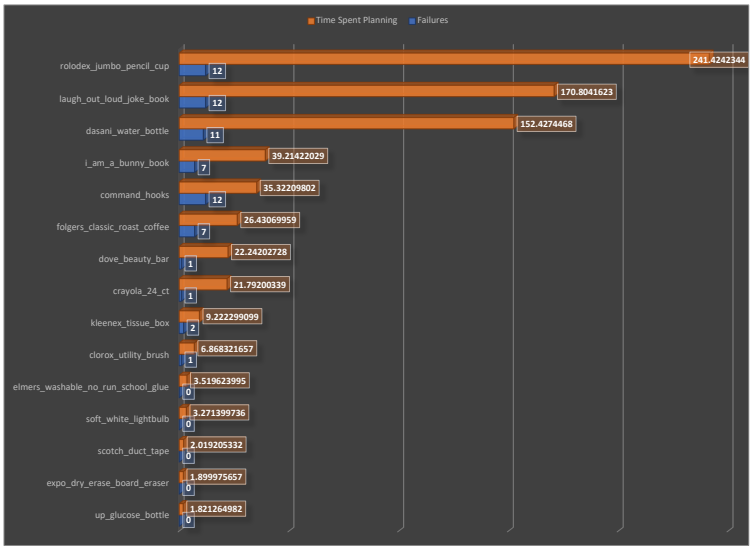


Figure A.11: Time Spent per Object. Roadmap Size: 100, Task Planner Mode: RR

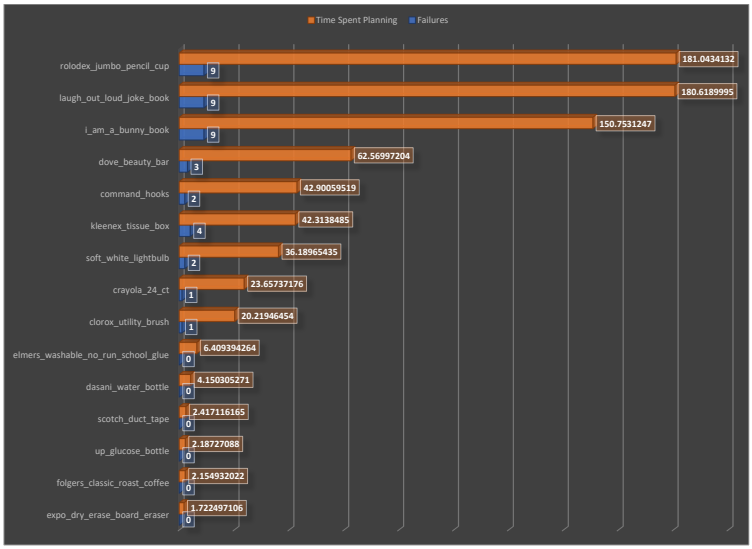


Figure A.12: Time Spent per Object. Roadmap Size: 30.000, Task Planner Mode: RR

Task: Picking, Task Planner Mode: Smart

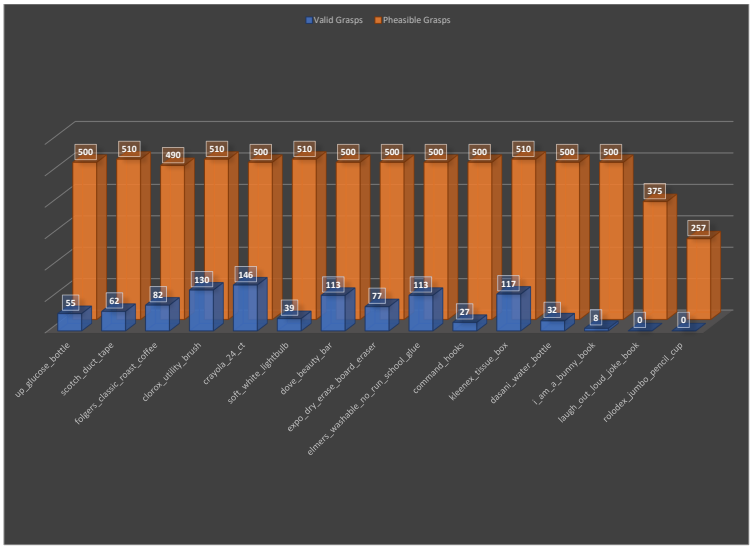


Figure A.13: Feasible vs Valid Grasps. Roadmap Size: 100, Task Planner Mode: Smart

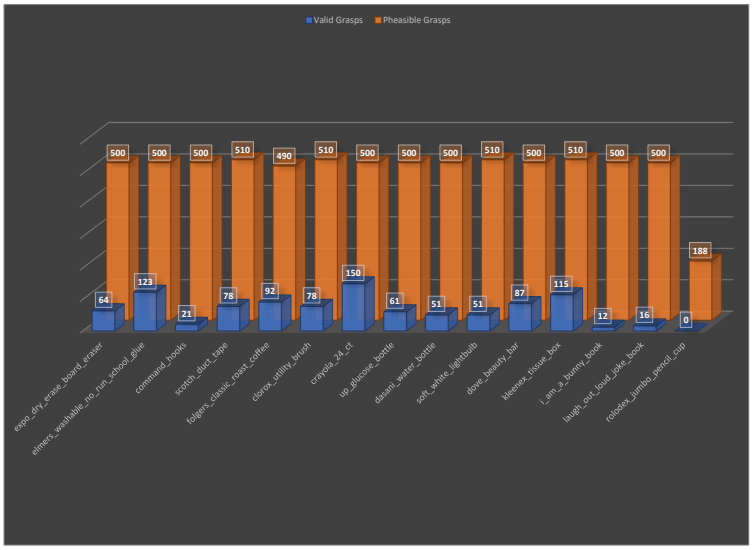


Figure A.14: Feasible vs Valid Grasps. Roadmap Size: 30.000, Task Planner Mode: Smart

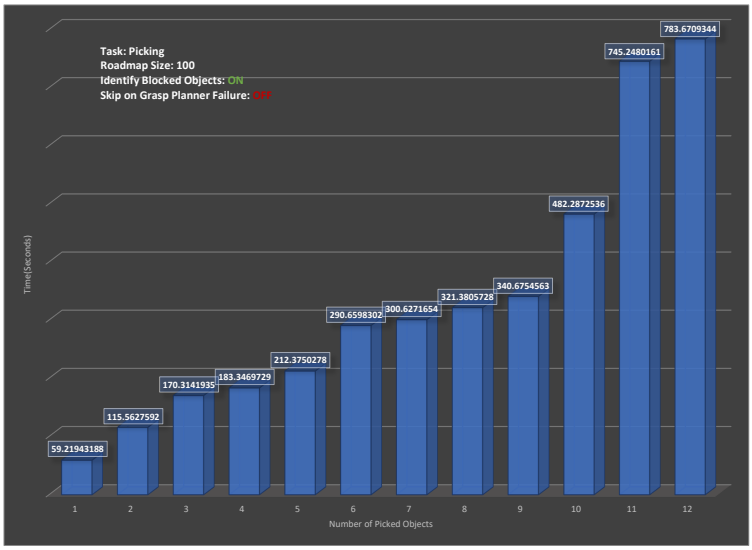


Figure A.15: Timestamp for each Pick. Roadmap Size: 100, Task Planner Mode: Smart

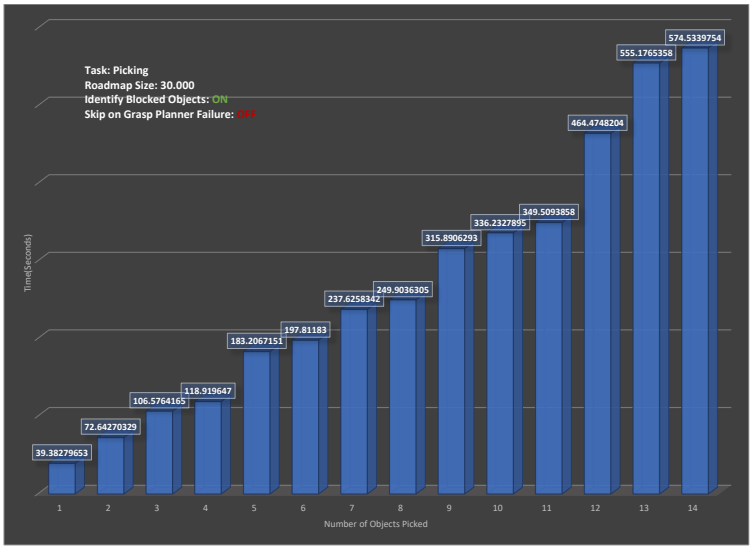


Figure A.16: Timestamp for each Pick. Roadmap Size: 30.000, Task Planner Mode: Smart

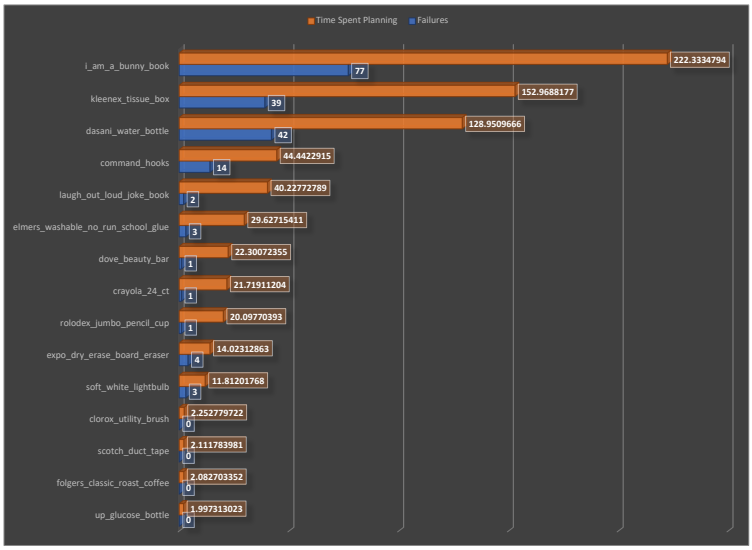


Figure A.17: Time Spent per Object. Roadmap Size: 100, Task Planner Mode: Smart

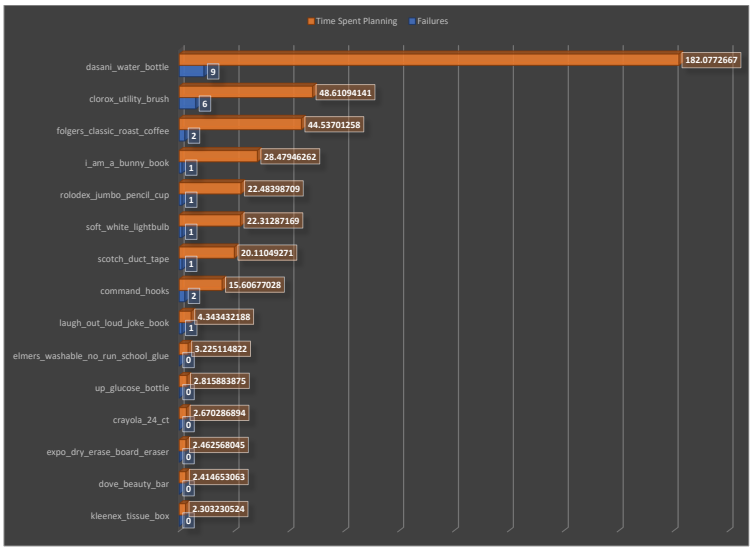


Figure A.18: Time Spent per Object. Roadmap Size: 30.000, Task Planner Mode: Smart

Task: Picking, Task Planner Mode: Smart Fast Cycle

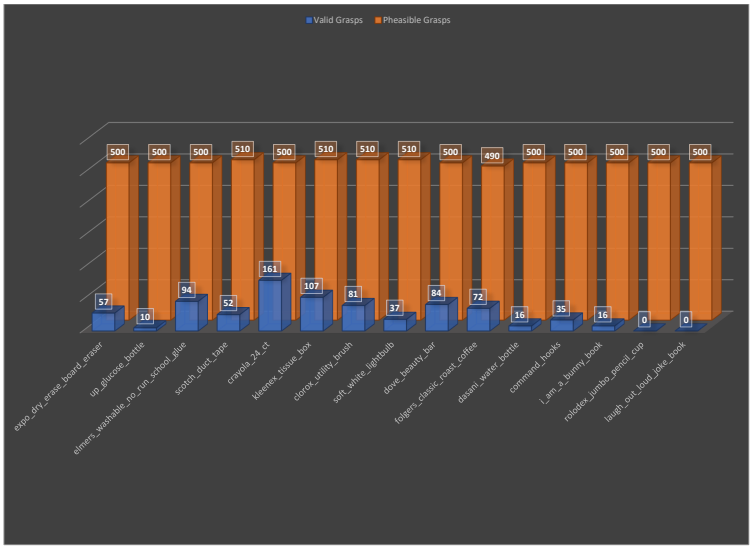


Figure A.19: Feasible vs Valid Grasps. Roadmap Size: 100, Task Planner Mode: SFC

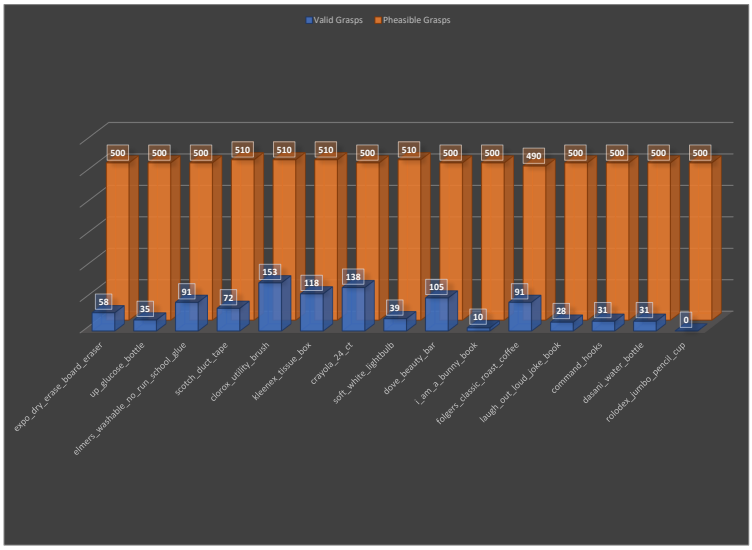


Figure A.20: Feasible vs Valid Grasps. Roadmap Size: 30.000, Task Planner Mode: SFC

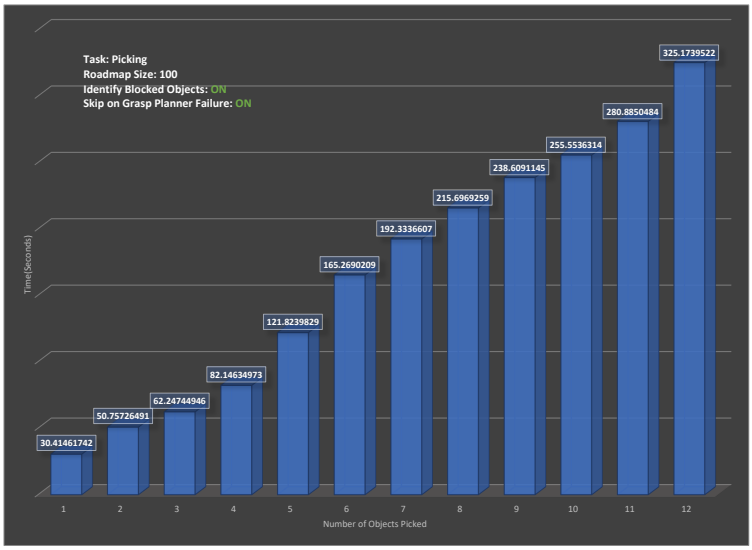


Figure A.21: Timestamp for each Pick. Roadmap Size: 100, Task Planner Mode: SFC

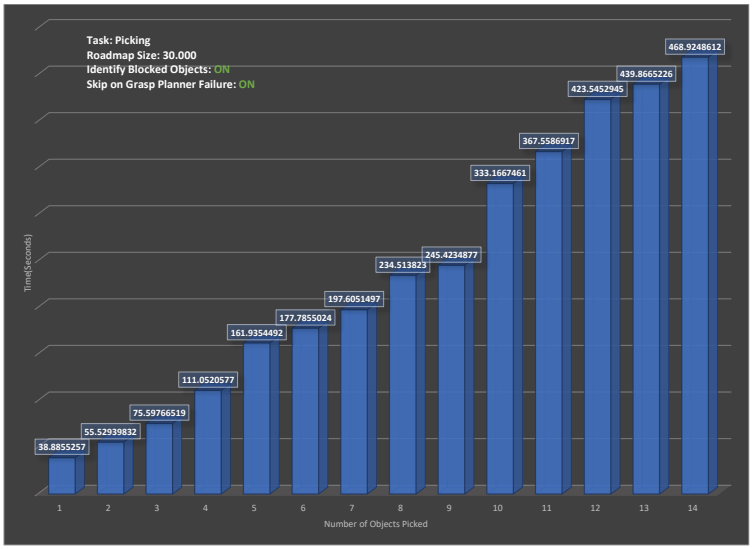


Figure A.22: Timestamp for each Pick. Roadmap Size: 30.000, Task Planner Mode: SFC

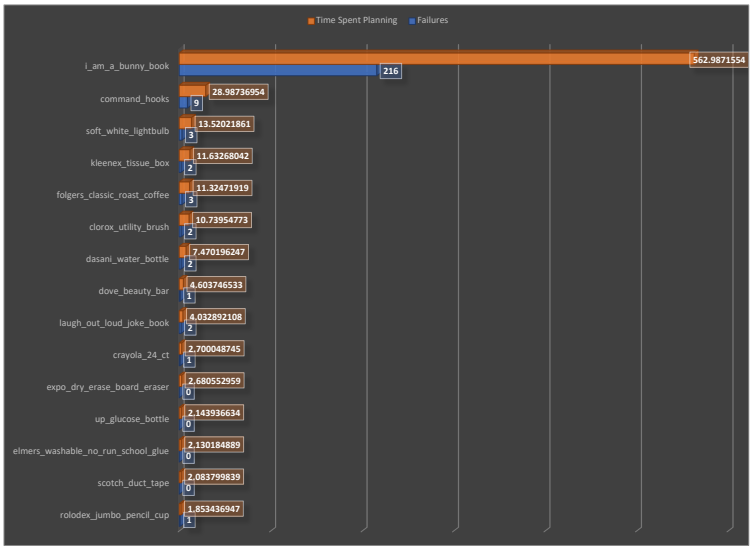


Figure A.23: Time Spent per Object. Roadmap Size: 100, Task Planner Mode: SFC

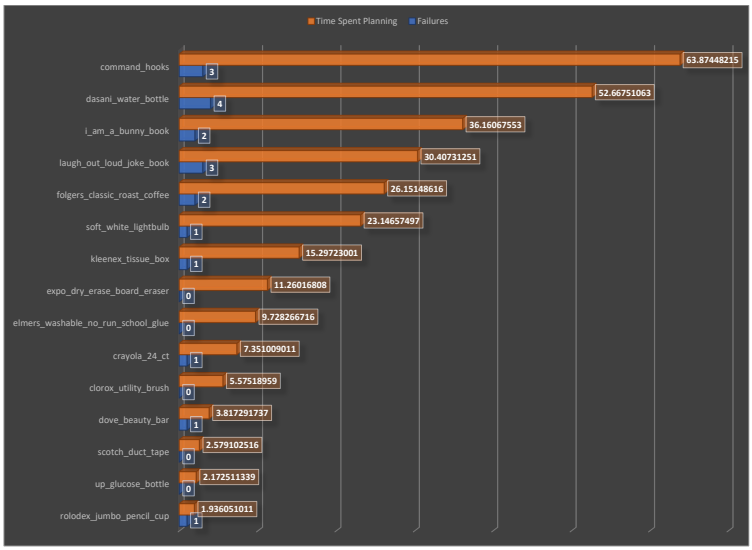


Figure A.24: Time Spent per Object. Roadmap Size: 30.000, Task Planner Mode: SFC

Task: Stowing, Task Planner Mode: Smart Fast Cycle

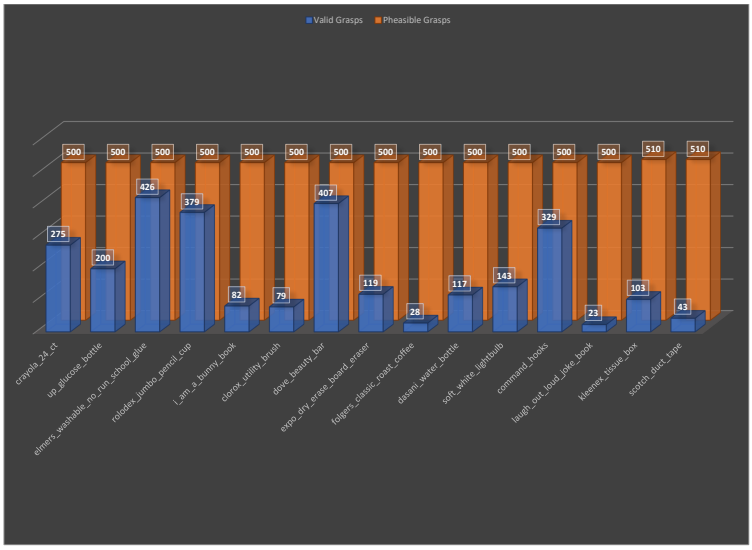


Figure A.25: Feasible vs Valid Grasps. Roadmap Size: 100, Task Planner Mode: SFC

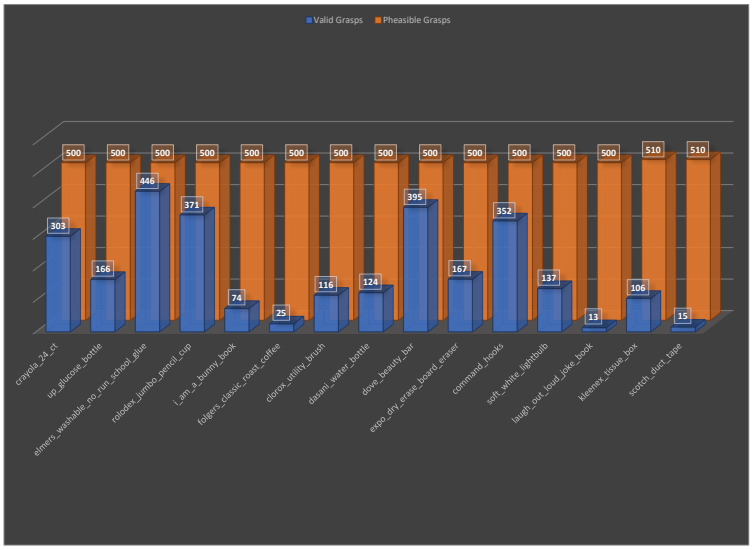


Figure A.26: Feasible vs Valid Grasps. Roadmap Size: 30.000, Task Planner Mode: SFC

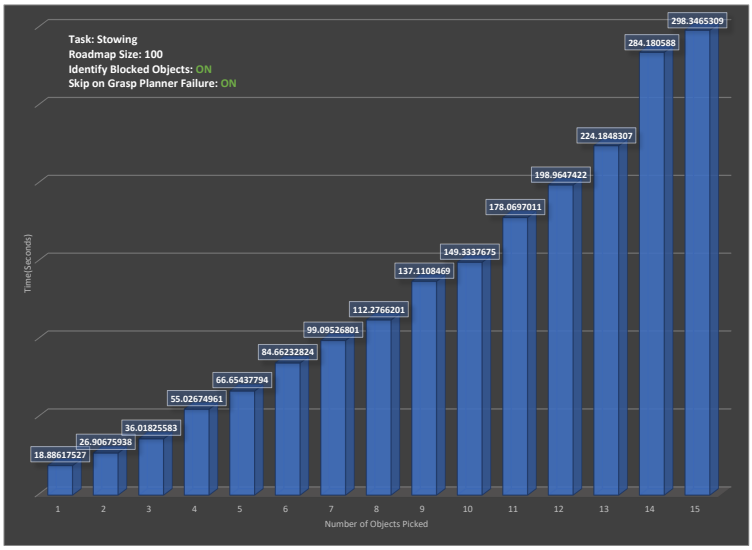


Figure A.27: Timestamp for each Pick. Roadmap Size: 100, Task Planner Mode: SFC

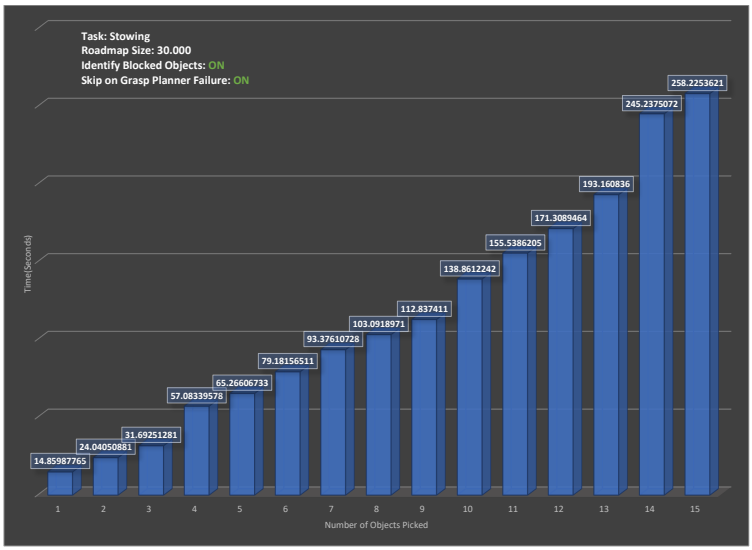


Figure A.28: Timestamp for each Pick. Roadmap Size: 30.000, Task Planner Mode: SFC

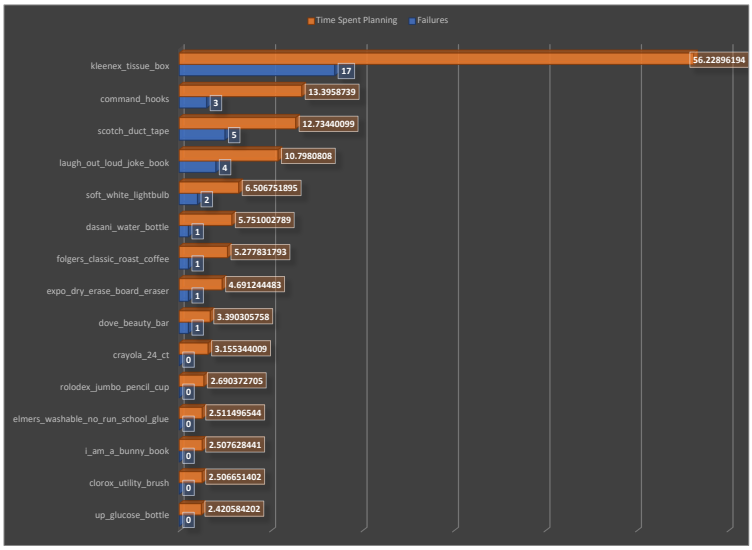


Figure A.29: Time Spent per Object. Roadmap Size: 100, Task Planner Mode: SFC

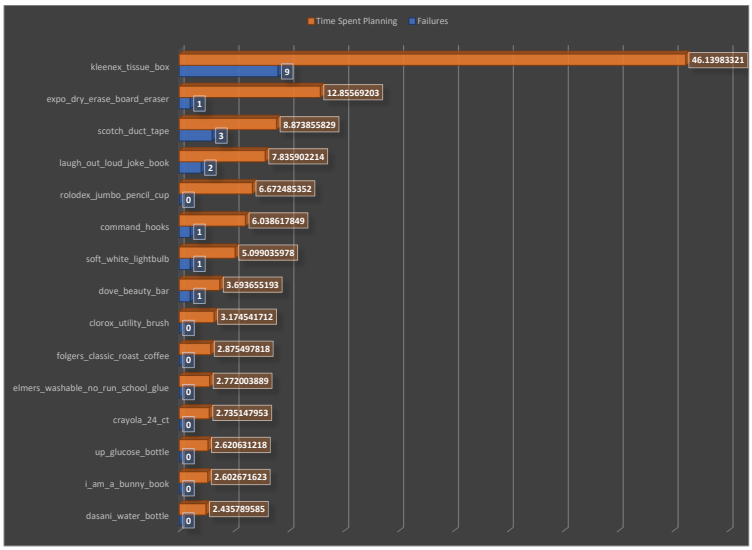


Figure A.30: Time Spent per Object. Roadmap Size: 30.000, Task Planner Mode: SFC

Acknowledgment of Previous Publications

[1] Zakary Littlefield, Shaojun Zhu, Hristiyan Kourtev, Zacharias Psarakis, Rahul Shome, Andrew Kimmel, Andrew Dobson, Alberto F De Souza, and Kostas E Bekris. Evaluating end-effector modalities for warehouse picking: A vacuum gripper vs a 3-finger underactuated hand. *In Automation Science and Engineering (CASE), 2016 IEEE International Conference* on, pages 11901195. IEEE, 2016.

[2] H. Kourtev, Z. Psarakis, K. E. Bekris. Evaluating End-Effector Modalities for Warehouse Picking. In *Northeast Robotics Colloquium 2016 (NERC)*, October 2016

[3] Z. Psarakis, H. Kourtev, A. Boularias and K.E. Bekris. Evaluating End-Effectors and System Integration for Warehouse Picking. In *2017 Warehouse Picking Automation Workshop 2017 (WPAW)*, May 2017

References

[1] P. R. Wurman, R. DAndrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," AI magazine, vol. 29, no. 1, p. 9, 2008.

[2] R. DAndrea, "Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead," IEEE Transactions on Automation Science and Engineering, vol. 4, no. 9, pp. 638639, 2012.

[3] P. Baker and Z. Halim, "An exploration of warehouse automation implementations: cost, service and flexibility issues," Supply Chain Management: An International Journal, vol. 12, no. 2, pp. 129138, 2007.

[4] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7):26–34, July 2008.

[5] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. Lessons from the amazon picking challenge. *CoRR*, abs/1601.05484, 2016.

[6] Carlos Hernandez, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, Mihai Morariu, Jihong Ju, Xander Gerrmann, Ronald Ensing, Jan van Frankenhuyzen, and Martijn Wisse. Team delft's robot winner of the amazon picking challenge 2016. *CoRR*, abs/1610.05514, 2016.

[7] D. Morrison, Adam W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, Anton Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, Christopher F. Lehnert, Ian D. Reid, Peter Corke, and Jürgen Leitner. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. *CoRR*, abs/1709.06283, 2017.

[8] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612, 2016.

[9] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, P. R. Wurman. "Lessons from the Amazon Picking Challenge," IEEE Transactions on Automation Science and Engineering. Also in arXiv:1601.05484 [cs.RO], 2016.

[10] Z. Littlefield, Shaojun Zhu, H. Kourtev, Z. Psarakis, R. Shome, A. Kimmel, A. Dobson, A. F. De Souza, and K. E. Bekris. Evaluating end-effector modalities for warehouse picking: A vacuum gripper vs a 3-finger underactuated hand. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1190–1195, Aug 2016.

[11] Z. Psarakis, H. Kourtev, A. Boularias and K. E. Bekris. Evaluating End-Effectors and System Integration for Warehouse Picking. In *2017 Warehouse Picking Automation Workshop 2017 (WPAW)*, May 2017

[12] Lael U. Odhner, Leif P. Jentoft, Mark R. Claffee, Nicholas Corson, Yaroslav Tenzer, Raymond R. Ma, Martin Buehler, Robert Kohout, Robert D. Howe, and Aaron M. Dollar "A Compliant, Underactuated Hand for Robust Manipulation," International Journal of Robotics Research, vol. 33(5), pp. 736-752, 2014.

[13] A. Rocchi, B. Ames, J. Li, and K. Hauser. "Stable Simulation of Underactuated Compliant Hands," IEEE Int'l. Conf. on Robotics and Automation (ICRA), 2016.

[14] R. Paolini, A. Rodriguez, S. Srinivasa, and M. T. Mason "A Data-Driven Statistical Framework for Post-Grasp Manipulation," The International Journal of Robotics Research (IJRR), Vol. 33, No. 4, April, 2014, pp. 600-615.

[15] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 14, IEEE, 2011.

[16] "G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library," OReilly Media, Inc., 2008.

[17] R. C. Willow Garage, "ORK: Object Recognition Kitchen," https:// github.com/wg-perception/object recognition core.

[18] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," European Conference on Computer Vision (ECCV), 2014.

[19] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient Response Maps for Real-Time Detection of Texture-Less Objects," IEEE Transactions on Pattern Analysis and Maschine Intelligence (TPAMI), vol. 34, no. 5, pp. 876888, 2012.

[20] R. c. Willow Garage, "Ecto a c++/python computation graph framework," http://plasmodic.github.io/ecto/.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., "Scikit-learn: Machine learning in python," The Journal of Machine Learning Research, vol. 12, pp. 28252830, 2011.

[22] Rennie, C., R. Shome, KE Bekris, and Ferreira A. De Souza. 2016. "A Dataset For Improved Rgbd-Based Object Detection And Pose Estimation For Warehouse Pick-And-Place," IEEE Robotics and Automation Letters (RA-L) [Also accepted to appear at the 2016 IEEE International Conference on Robotics and Automation (ICRA)] 1(2): 1179 - 1185.

[23] Pauwels, Karl and Kragic, Danica "SimTrack: A Simulation-based Framework for Scalable Real-time Object Pose Detection and Tracking," IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany, 2015.

[24] C. Mitash, K. E. Bekris, A. Boularias, "A Self-supervised Learning System for Object Detection using Physics Simulation and Multi-view Pose Estimation," arXiv preprint arXiv:1703.03347 , 2017

[25] Leitner J, Frank M, Forster A, Schmidhuber J, "Reactive reaching and grasping on a humanoid: Towards closing the action-perception loop on the iCub," Proceedings of the 2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Volume 1 p102-109, 2014.

[26] A. Bicchi and V. Kumar, "Robotic Grasping and Contact: A Review," in ICRA, 2000.

[27] A. Sahbani, S. El-Khoury, and P. Bidaud, "An Overview of 3D Object Grasp Synthesis Algorithms," RAS, vol. 60, no. 3, pp. 326336, 2012.

[28] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," HUMANOIDS, Seoul, Korea, November 2015.

[29] A. Miller and P. Allen, "GraspIt!: A Versatile Simulator for Robotic Grasping," IEEE RAM, vol. 11, no. 4, pp. 110122, 2004.

[30] A. Miller and P. Allen. GraspIt!: A Versatile Simulator for Robotic Grasping. *IEEE RAM*, 11(4):110–122, 2004.

[31] M. Ciocarlie and P. Allen. A Design and Analysis Tool for Underactuated Compliant Hands. In *IROS*, 2009.

[32] M. Roa and R. Surez. Grasp Quality Measures: Review and Performance. *Autonomous Robots*, 38:65–88, 2015.

[33] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka. Human-guided Grasp Measures Improve Grasp Robustness on Physical Robot. In *ICRA*, 2010.

[34] Z. Li and S. S. Sastry. Task-oriented Optimal Grasping by Multi-fingered Robot Hands. In *Journal of Robotics and Automation*, pages 32–44, 1988.

[35] C. Ferrari and J. Canny. Planning Optimal Grasps. In *ICRA*, 1992.

[36] S. Liu and S. Carpin. A Fast Algorithm for Grasp Quality Evaluation Using the Object Wrench Space. In *CASE*, 2015.

[37] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case-study," Journal of Software Engineering in Robotics, Special issue on Best Practice in Robot Software Development, vol. 5, no. 1, pp. 316, 2014.

[38] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, vol. 79, 2008.

[39] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2014.

[40] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," Robotics: Science and Systems, vol. 9, pp. 110, Citeseer, 2013.

[41] Kimmel, A, A Dobson, Z Littlefield, A Krontiris, J Marble, and KE Bekris. 2012. "Pracsys: An Extensible Architecture For Composing Motion Controllers And Planners," Simulation, Modeling and Programming for Autonomous Robots (SIMPAR), Tsukuba, Japan.

[42] Littlefield, Z, A Krontiris, A Kimmel, A Dobson, R. Shome, and KE Bekris. 2014. "An Extensible Software Architecture For Composing Motion And Task Planners," International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Bergamo, Italy.

[43] S. Kumar and L. Behera, "Implementation of a neural network based visual motor control algorithm for a 7 DOF redundant manipulator," World congress on computational intelligence (WCCI), Hong Kong, June 1- 6, 2008.

[44] H. Kourtev, 2018. "A robust soft and vacuum hybrid end-effector and compliant arm for picking in clutter.", M.S. Thesis, Rutgers University, https://rucore.libraries.rutgers.edu/rutgers-lib/56044/