Institut für Informatik
Technische Universität München

Action-Related Places for Mobile Manipulation

Dissertation

Andreas Fedrizzi

TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik

Lehrstuhl für Bildverstehen und wissensbasierte Systeme

Action-Related Places for Mobile Manipulation

Andreas Fedrizzi

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Darius Burschka
Prüfer der Dissertation:	1.	Prof. Michael Beetz, PhD
	2.	Prof. Rachid Alami, LAAS/CNRS, Toulouse, Frankreich

Die Dissertation wurde am 09.11.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 27.06.2011 angenommen.

Abstract

Today's autonomous robots cannot perform everyday manipulation tasks with human dexterity and flexibility. One of the main reasons is that even seemingly simple tasks such as picking up an object turned out to be surprisingly difficult decision making problems. To pick up an object the robot must decide where to stand in order to pick up the object, which hand(s) to use, how to reach for the object, which grasp type to apply, where to grasp, how much grasp force to apply, how and where to hold the object, how to lift the object, and how much force to apply to lift it. The decision problems are even more complex because many decisions depend on task context, which requires the robot to take many factors into account in order to achieve the best possible performance.

Humans deal with this complexity by selecting and parameterizing their actions based on predicted action consequences. For example, when considering the problem of choosing a location from which an object should be grasped, then humans prefer locations from which the grasping action will likely succeed. This dissertation thesis presents the framework of Action-Related Places, which is a novel approach for computing manipulation places that maximize expected utility. The framework of Action-Related Places advances the state of the art by: (1) Introducing a compact and precise representation of manipulation places; (2) Learning an internal model of manipulation place through experience-based learning; (3) Enabling online-reasoning about manipulation places with ARPlace probability distributions.

Manipulation places for robot positioning are represented by internal models that are learned knowledge of places that lead to successful manipulation. They implicitly take the robot's skills into account. As a result, the proposed manipulation places are not only kinematically promising, but have led to successful manipulation in actual manipulation tasks. We call the internal model for robot positioning Generalized Success Model and use experience-based learning to develop it. A robot performs several exemplary manipulation actions and stores whether the action succeeded or failed. Support vector machines and point distribution models are used to generalize over the gathered data. We chose point distribution models as the basis for Generalized Success Models because they are compact, can be queried quickly, and it is possible to take state estimation uncertainties into account.

The Generalized Success Model is used online in order to compute Action-Related Places (ARPLACEs), which map base positions to the probability that the robot will be able to successfully perform the manipulation task when it is executed from there. ARPLACE probability distributions are updated iteratively when new sensor information arrives. Moreover, taking state estimation uncertainties into account makes the resulting manipulation place more robust than approaches that assume ground truth data. Because ARPLACEs capture grasp success probability for all places, it is possible to find optimal manipulation places. The principle of optimality can also be met when there are multiple, potentially conflicting task goals by utilizing decision theory.

The framework of Action-Related Places was fully implemented and evaluated in the Gazebo simulator. Evaluations showed that manipulation places that are proposed by the ARPLACE framework are better in a statistically significant manner than manipulation places that were proposed by other place-finding strategies. Manipulation places from the ARPLACE framework are especially more robust when the robot is not able to precisely locate the target object. Applying transformational planning reduced the average execution time from 48 seconds to 32 seconds when multiple objects had to be grasped.

Kurzfassung

Menschen sind heutzutage autonomen Robotern bei der Verrichtung von Manipulationsaufgaben im Alltag weit überlegen. Einer der Hauptgründe ist, dass sogar einfache Aufgaben wie ein Objekt auf einem Tisch zu greifen sich als Sequenz erstaunlich komplizierter Entscheidungsprobleme herausgestellt haben. Um ein Objekt zu greifen muss sich der Roboter überlegen wo er sich für den Greifvorgang positionieren soll, welche Hand er benutzen soll, ob er nicht sogar beide Hände benötigt, wie und wo das Objekt am besten zu Greifen ist, wieviel Kraft beim Griff angewendet werden sollte, und wieviel Kraft beim Hochheben sinnvoll ist. Die Entscheidungsprobleme sind sogar noch komplizierter weil bei vielen Entscheidungen der Aufgabenkontext eine wichtige Rolle spielt und berücksichtigt werden muss.

Menschen werden mit dieser Komplexität fertig, indem sie Aktionsentscheidungen und Aktionsparametrisierungen auf Basis von prognostizierten Aktionskonsequenzen treffen. Wenn Menschen sich beispielsweise für einen Platz entscheiden müssen von dem aus sie ein Objekt greifen, dann präferieren sie Orte von denen aus die Greifaktion vermutlich erfolgreich sein wird. Die vorliegende Dissertation stellt das *Action-Related Places* Framework vor, welches ein neuer Ansatz ist um Plätze zu finden die den erwarteten Nutzen einer Manipulationsaktion maximieren. Das Action-Related Places Framework erweitert den aktuellen Stand der Forschung wie folgt: (1) Es führt eine kompakte und präzise Repräsentation von Manipulationsplätzen ein; (2) Es entwickelt interne Modelle von Manipulationplätzen durch erfahrungsbasiertes Lernen; (3) Zur Laufzeit ermöglicht es logisches Schliessen über Manipulationsplätze anhand von ARPLACE Wahrscheinlichkeitsverteilungen.

Manipulationsplätze zur Roboterpositionierung werden in internen Modellen repräsentiert. Die internen Modelle werden anhand von Wissen gelernt, welche Manipulationsplätze für bestimmte Manipulationsaufgaben zum Erfolg führten und welche nicht und spiegeln dadurch implizit die Hardware-Fähigkeiten des Roboters wieder. Als Konsequenz sind die berechneten Manipulationsplätze nicht Orte welche nur auf einer kinematischen Ebene gut sein müssten, sondern Manipulationsplätze für die sich in der Vergangenheit gezeigt hat, dass sie in tatsächlichen Manipulationsaufgaben erfolgreich sind. Wir nennen diese internen Modelle zur Roboterpositionierung Generalized Success Models und verwenden erfahrungsbasiertes Ler-

nen um sie zu akquirieren. Dazu führt ein Roboter einige Manipulationsaktionen aus und speichert ob die jeweilige Aktion erfolgreich war oder nicht. Support Vector Machines und Point Distribution Models werden verwendet um über die gewonnenen Daten zu generalisieren. Wir haben uns für Point Distribution Models als Basis für das Generalized Success Model entschieden, da sie kompakt sind, schnell abgefragt werden können, und es möglich ist Unsicherheiten bezüglich der aktuellen Zustandsschätzung des Roboters zu berücksichtigen.

Das Generalized Success Model wird zur Laufzeit benutzt um Action-Related Places (kurz: ARPLACEs) zu berechnen, welche für beliebige Manipulationsplätze die Wahrscheinlichkeit widerspiegeln, daß die Manipulationsaufgabe erfolgreich ausgeführt werden kann, wenn sie von dort gestartet wird. ARPLACE Wahrscheinlichkeitsverteilungen werden iterativ aktualisiert wenn neue Sensorinformation eintrifft. Darüberhinaus werden Unsicherheiten bezüglich der aktuellen Zustandsschätzung des Roboters explizit berücksichtigt. Dieses Vorgehen macht die Bestimmung von Manipulationsplätzen erheblich robuster als Verfahren, die von ground truth Daten ausgehen. Dadurch, dass ein Action-Related Place die Greifwahrscheinlichkeit für alle Manipulationsplätze berechnet, lassen sich Aussagen bezüglich der Güte von Manipulationsplätzen treffen und es ist möglich optimale Manipulationsplätze zu bestimmen. Wir verwenden Entscheidungstheorie um das Prinzip der Optimalität auch dann zu wahren, wenn in einer Manipulationsaufgabe mehrere, sich potentiell widersprechende Ziele verfolgt werden.

Das ARPLACE Framework wurde komplett implementiert und im Gazebo Simulator umfangreich evaluiert. Es konnte gezeigt werden, dass Manipulationsplätze, die vom Action-Related Places Framework vorgeschlagen wurden, statistisch signifikant besser sind als Manipulationsplätze die von anderen Strategien vorgeschlagen wurden. Insbesondere sind Manipulationsplätze des ARPLACE Framework deutlich robuster, wenn der Roboter das Zielobjekt nicht exakt lokalisieren kann. Die Anwendung eines Transformationsplaners ermöglichte es zudem die Ausführungszeit von durchschnittliche 48 Sekunden auf 32 Sekunden zu reduzieren wenn mehrere Objekte gleichzeitig manipuliert werden mussten.

Acknowledgements

I want to thank everybody who supported me during my research and during the writing of this thesis. First, I want to thank my advisor Prof. Michael Beetz for giving me the opportunity to work in the exciting field of robotics research. In the Intelligent Autonomous Systems Group he managed to bring together a group of talented people that are passionate about robotics. To work with them was inspiring and the numerous discussions at the chair were an important aspect for the success of this thesis. In this context I want to especially thank Dr. Freek Stulp and Dominik Jain. Additionally I want to thank Lorenz Mösenlechner, Moritz Tenorth, Dr. Jan Bandouch, and Dr. Franziska Zacharias for the fruitful collaboration during our joint publications. Furthermore I am grateful to Prof. Rachid Alami for being my second advisor. With his input and stimulating discussions he helped to strengthen the thesis.

It is a challenging task to build and maintain autonomous robots. Therefore I am grateful to all the people that contributed to making the B21 robot a working mobile manipulation platform. Special thanks goes to Alexis Maldonado, Federico Ruiz, Ingo Kresse, and Ulrich Klank. Finally I want to thank Dr. Jan Bandouch for being a pleasant room mate over years and my family and friends for their past and ongoing support.

This work was funded by the Technische Universität München, the CoTeSys cluster of excellence, part of the Excellence Initiative of the DFG, and the DFG ActAr project. I owe to Prof. Giorgio Metta and the Italian Institute of Technology for being able to work there as a guest researcher. Becoming familiar with the robot platform iCub was a valuable experience.

Contents

List of Figures

List of Algorithms

List of Symbols

Symbols related to positions and pose estimations

x, y, and z	x-, y-, and z-coordinate
$\phi, \theta,$ and ψ	Roll-, pitch- and yaw-angle
$\langle x, y \rangle$	2D position
$\langle x, y, \psi \rangle$	2D pose
$\langle x, y, z \rangle$	3D position
$\langle x, y, z, \phi, \theta, \psi \rangle$	3D pose

The above symbols are used with the following suffixes

$\circ_{\text{rob}}$	ground truth coordinate/angle of the robot ($\circ \in \{x, y, \psi\}$)
$\circ_{\text{obj}}$	ground truth coordinate/angle of the object ($\circ \in \{x, y, z, \phi, \theta, \psi\}$)

Furthermore, we define the following prefixes

$\Delta_\circ$	coordinate/angle with respect to relative feature space
$\mu_\circ$	mean of the coordinate/angle as estimated by robot
$\sigma_\circ$	standard deviation of coordinate/angle as estimated by robot

Examples

x_{rob}: true position of the robot with respect to the x-axis of the world frame

$\Delta\psi_{\text{obj}}$: true yaw angle of the object with respect to the x-axis of the RFS

$\mu_{x_{\text{rob}}}$: robot's estimation of its base position with respect to the x-axis of the world frame

$\sigma_{\Delta\psi_{\text{obj}}}$: robot's estimated standard deviation of the object's ψ-angle with respect to the RFS

Symbols related to ARPLACE distributions

(x, y)	Grid cell at index x and y
$c(x, y)$	Center position of grid cell (x,y)
$f(x, y)$	Function that maps grid cells to its ARPLACE value
f^*	$\max_{(x,y)} f(x, y)$: Maximum ARPLACE value among all grid cells
(x^*, y^*)	$\arg\max_{(x,y)} f(x, y)$: Grid cell with maximum ARPLACE value
p_O	Obstacle probability distribution
$p_O(x, y)$	Probability that grid cell (x, y) is occupied

$p_U(x, y)$	Probability that grid cell (x, y) is unoccupied
p_R	Repeller probability distribution
$p_R(x, y)$	Probability value of grid cell (x, y) of repeller distribution
p	ARPlace probability distribution
p_{rl}	ARPlace probability distribution for grasping two objects at once; obj_1 with the right arm and obj_2 with the left arm
p_{lr}	ARPlace probability distribution for grasping two objects at once; obj_1 with the left arm and obj_2 with the righ arm
$p_S(x, y)$	Basic grasp success probability at grid cell (x, y)
$p(x, y)$	Combined grasp success probability at grid cell (x, y)
w_S	Importance of success
$u_S(x, y)$	Utility of success at grid cell (x, y) ($= p(x, y) \cdot u_S$)
p^*	$\max_{(x,y)} p(x, y)$: maximum grasp success probability among all grid cells
(x_{p^*}, y_{p^*})	$\arg\max_{(x,y)} p(x, y)$: grid cell with maximum grasp success probability
$t(x, y)$	Estimated travel time to grid cell (x, y)
w_T	Importance of time
$u_T(x, y)$	Utility of time at grid cell (x, y)
t^*	$\min_{(x,y)} t(x, y)$: minimum estimated travel time among all grid cells that have a grasp success probability of more than 0%
(x_{t^*}, y_{t^*})	$\arg\min_{(x,y)} t(x, y)$: grid cell with minimum estimated travel time
$e(x, y)$	Consumed energy for performing task from grid cell (x, y)
$u_E(x, y)$	Utility of saving energy at grid cell (x, y)
w_E	Importance of saving energy
u	ARPlace utility distribution
$u(x, y)$	Expected utility at grid cell (x, y)
u^*	$\max_{(x,y)} u(x, y)$: maximum utility value among all grid cells
(x_{u^*}, y_{u^*})	$\arg\max_{(x,y)} u(x, y)$: grid cell with maximum utility value

List of Abbreviations

ARPLACE	Action-Related Place
ARPLACE$_{\mathrm{PROB}}$	Action-Related Place based on grasp success probability
ARPLACE$_{\mathrm{UTIL}}$	Action-Related Place based on utility
BGSP	Basic Grasp Success Probability
CGSP	Combined Grasp Success Probability
DOF	Degrees Of Freedom
GSM	Generalized Success Model
GSP	Grasp Success Probability
IK	Inverse Kinematics
MCMC	Markov Chain Monte Carlo
PDM	Point Distribution Model
RFS	Relative Feature Space
SLAM	Synchronous Localization and Mapping
SVM	Support Vector Machine

CHAPTER 1

Introduction

In their roadmap for US robotics, Hollerbach et al. (2009) consider robotics as a "key economic enabler". The authors acknowledge industrial robots to "provide increased accuracy and throughput for particular, repetitive tasks, such as welding, painting, and machining, in hazardous, high volume manufacturing environments", but believe that "the applications for such first generation robotics solutions have proven to be relatively narrow and largely restricted to static indoor environments, due to limitations in the enabling technology". It is claimed that second generation robots will be applicable to a whole new range of applications such as agile manufacturing, logistics, medicine, and healthcare. "Owing to the inexorable aging of our population, the emergence of such a next generation, robotech industry will eventually affect the lives of every American and have enormous economic, social, and political impact on the future of our nation". Hollerbach et al. (2009) believe that "human-like dexterous manipulation" is one of the "critical capabilities" for realising second generation robots. Only robots that are able to physically interact with their environment are able to perform sophisticated tasks like cooking a meal, cleaning the floor, or setting a table.

Mobile manipulation, however, is one of the problems that are very easy for human people to perform but hard to reproduce artificially. A human person can interact with its environment without even consciously thinking about it. But when actually asked how the interaction was exactly done, then it is hard to articulate this in a comprehensive manner. The problem is similar to natural language. Human people are able to speak without consciously thinking about it. However, computers did not yet manage to acquire human speaking skills. So far, no computer system has passed the Turing test where a human person talks to a computer without seeing it. If the human person can not tell that the dialog partner is a computer, then the computer must have really understood the principle of human speech. The Turing test can be seen as a benchmark for speaking skills. Rosenbaum et al. (2006) adapts this idea to a Turing test of robot action and proposes it as a benchmark test of motor intelligence. "If

[Elana] watched a robot dancing and thought it was a human being, she could say the designer of the dancing robot had captured, and truly *understood*, the control of dance."

In a final report on a NSF/NASA workshop on autonomous mobile manipulation, Brock and Grupen (2005) attribute "significant economical, societal, and scientific importance" to the challenge of creating "robotic agents capable of performing physical work in unstructured and open environments". The workshop participants identified reserach areas such as Mobility, Representing Objects and Environments, Grasping and Dexterous Manipulation, Perception, and Human-Robot Interaction among the most important in the context of autonomous mobile manipulation.

Kemp et al. (2007) present insights from an international RSS workshop about "Manipulation for Human Environments". Several important lines of research are identified in this roadmap paper: Perception, Learning, Working with People, Platform Design, and Control. These topics make mobile manipulation in human environments a research challenge for the upcoming decades.

Even seemingly simple tasks such as picking up an object from a table requires complex decision making. To pick up an object the robot must decide where to stand in order to pick up the object, which hand(s) to use, how to reach for the object, which grasp type to apply, where to grasp, how much grasp force to apply, how and where to hold the object, how to lift the object, and how much force to apply to lift it. The decision problems are even more complex because many decisions depend on task context, which requires the robot to take many factors into account in order to achieve the best possible performance.

1.1 Problem Statement

The decision problem that we will examine in this thesis is to which base positions robot should move in order to perform manipulation actions. Figure 1.1 depicts an exemplary scenraio where a B21r mobile robot has to grasp a cup on a table.

A trivial approach to solve this task is to go to a position such that the target of manipulation is well in reach. However, a more careful look at the question raises some serious issues. What is a good place in the context of an intended manipulation action? Does well-in-reach always imply that the target object can really be reached given the hardware and control software of the robot? What if hardware parameters such as friction in the joints change over time? How can the robot take into account uncertainties about its self-localization and the estimated position of the target object? And finally, can a robot that is able to perform the task in a European kitchen be successful in a Japanese kitchen? Simply navigating to a location that is close to

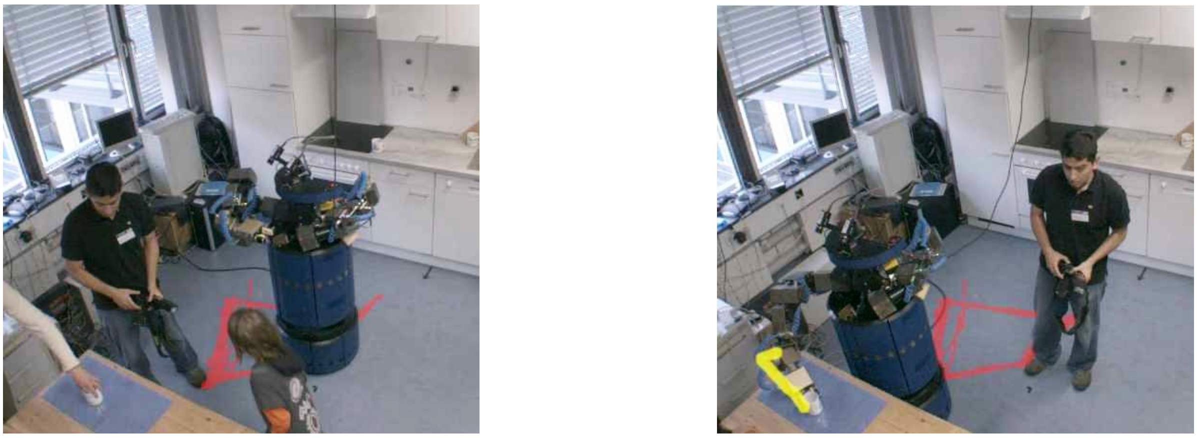


FIGURE 1.1 A reach and grasp trajectory performed by the B21r TUM kitchen assistant during a public demonstration. Left: Robot navigates to table so that the cup is well in reach. Right: Arm reaches for cup. Note that the operator is holding a camera, not a remote control.

the target object is not enough because the robot not only has to account for its navigation and manipulation skills, but also requires knowledge of the interactions and coupling between them.

When the vision system revises its hypothesis for the target object's pose, then the optimal base position may change. When a second object is detected, then grasping both objects at once may save time. And when an obstacle is detected that blocks promising base positions, then alternative solutions have to be found. Moreover, robots have to deal with the problem that their skills may change over time. Motors may loose power due to aging, joint friction may increase over time, and hardware as well as software components may be updated.

1.1.1 Cognitive Motor Control

For humans all these decisions and online-adjustments seem to be simple. A human person can interact with its environment without even consciously thinking about it. Trying to understand the human's manipulation system may help to advance robotic manipulation skills. Jordan and Wolpert (1999) identify four important areas of human motor control. Motor planning, the representation of motion with internal models, state estimation, and using multiple internal models for motor control. They claim that "it is important to emphasize that an internal model is a form of knowledge about the environment". As we shall see later, Generalized Success Models that capture promising base positions for manipulating objects are internal models for robot positioning.

According to Wolpert and Kawato (1998) the internal model of the human motion system consists of multiple paired forward and inverse models. Inverse models are used for controlling motions and forward models are used to predict the outcome of motor control inputs. Forward and inverse models are tightly coupled in the acquisition (motor learning) and execution phase. This model suggests that human motor control is a closed loop system and that action and prediction is tightly coupled.

Shadmehr and Mussa-Ivaldi (1994) investigated how the central nervous system learns reaching movements in the presence of externally imposed forces. They found that subjects initially perform trajectories that differ from their usual paths and velocity profiles. But over time the subjects adapted to the force fields and returned to natural movement. Interesting is that "subjects modeled the force field by a combination of computational elements" and Shadmehr and Mussa-Ivaldi (1994) explicitly state that "adaptation was not via composition of a look-up table". This also speaks in favor of multiple forward and inverse models.

In robotics, a system that consists of a closed perception-action loop is the one presented by Katz and Brock (2008). It deals with the problem of building kinematic models of a priori unknown objects. Therefore, the robot watches and analyzes its interactions with the world and is able to reveal information that would otherwise remain hidden. They call their approach for coupling perception and action *interactive perception*.

1.1.2 Bayesian Modeling and Bayesian Brain

Bayesian models are a current technique to understand human intelligence in terms of rational probabilistic inference. Griffiths et al. (2008) present a thorough overview of this topic and study the question why people are so good in domains such as how the mind infers the intrinsic properties of a object, or how children infer the rules of grammar. "In each of these cases, the available data severely underconstrain the inferences that people make, and the best the mind can do is to make a good guess, guided - from a Bayesian standpoint - by prior probabilities about which world structures are most likely a priori".

Another concept that is closely related to bayesian modeling is the bayesian brain. Knill and Pouget (2004) broadly cover this topic and present the bayesian coding hypothesis: "the brain represents sensory information probabilistically, in the form of probability distributions". This seems to be a promising concept for finding base positions for manipulation, as well. Compute the probability that the manipulation action will succeed from certain base positions and choose the one that maximizes this grasp success probability.

The research areas of cognitive motor control, bayesian modeling and bayesian brain suggest that the human's ability to robustly act in unconstrained environments is grounded in several abilities. Among them are the abilities to take state estimation uncertainty into account, to dynamically parameterize actions by dynamically adapting to the context of the current task, and to be able to make elaborate decisions even if the the task is underconstrained. These findings suggest that pre-programmed solutions will not enable robots to autonomously act in unconstrained environments.

The goal is to develop compact control programs that are able to handle the process of complex decision making and can adapt to complex and changing environments in a natural and cognitive way.

1.1.3 ARPlaces as Cognitive Approach to find Manipulation Places

In this thesis we present an example of such control programs, namely finding promising base positions for robots that want to perform manipulation tasks. We call such promising base positions *places* or *manipulation places*. Manipulation places are represented by *Action-Related Places* (ARPlaces). The following Lisp code is the robot's internal representation of a place description. It specifies an entitity of the category ARPlace for successfully picking up an object.

```
(the ARPlace
     (task   (a task (task-action     pick-up)
                     (objectActedOn (a object     on table)))))
```

This specification of a manipulation place is abstract and symbolic but not yet effective as it does not tell *where* the robot should exactly position itself in order to perform the pick up task. Instead of inferring the right position for picking up a target object and setting the respective parameter, our robot control system instantiates an ARPlace for this task: the position for picking up the object. Having the respective concept instantiation represented explicitly during the course of action enables the robot to reconsider and reevaluate the decision whenever new evidence arrives.

But how is an ARPlace represented? We considered *grasp success probability* to be appropriate, which assigns to every base position the probability with which the manipulation action will succeed when it is performed from there. To illustrate this point, consider the scenario depicted in Figure 1.2, in which the robot has to pick up cups from a table.

Consider the place where the robot should position itself in order to pick up the cups as an ARPlace. When entering the room the robot sees a cup on the table but can only localize it inaccurately and therefore the specification of the place from where to pick up the object is uncertain (step 1). As the robot moves closer, its position estimate of the cup becomes more precise and therefore the place becomes better defined (indicated by deeper green in step 2).

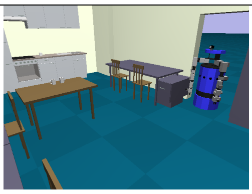
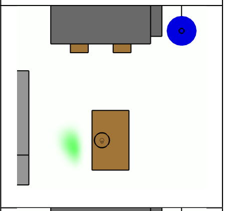
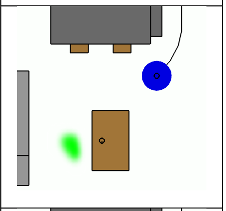
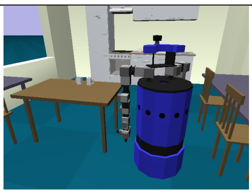
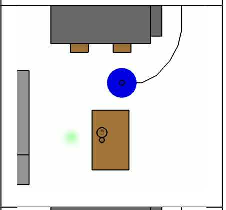
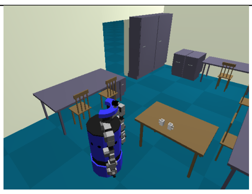
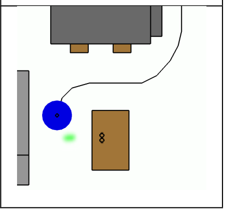
Step	Scene	ARPLACE
1		
2		
3		
4		

FIGURE 1.2 The dynamic update of an Action-Related Place in which the robot approaches a table in order to pick up a cup. The images on the right side visualize the robot's current information about its environment. The black circles depict uncertainty of the robot's estimations about its own position (black circle in blue robot shape), and about the cup positions (black circles around cups). Bigger circles represent higher uncertainty. The green area is the ARPLACE. Deeper green marks higher grasp success probability.

Upon noticing a second cup on the table in step 3, the robot checks whether there is a manipulation place from which it can reach both cups, and updates the ARPLACE accordingly. In the final step, the localization of the second cup becomes more precise and therefore the manipulation place is better defined (deeper green). This scenario shows that ARPLACEs provide context-adapted manipulation places throughout the episode, and enable online updating of these places.

1.2 Scientific Contributions

This dissertation thesis presents the framework of Action-Related Places. ARPLACEs solve the problem of *finding optimal places for performing subsequent manipulation actions*. They are therefore relevant to researchers in the robotics community that perform mobile manipulation. The main contributions of the framework of Action-Related Places are as follows.

1.2.1 Representation of Places for Successful Manipulation

Good manipulation places significantly depend on the skills of a robot such as the robot's kinematics, its controllers, as well as motion planning systems that are used. We propose Generalized Success Models as a compact and precise internal model of manipulation places that explicitly takes the robot's hardware skills into account. Generalized Success Models can be seen as compiled knowledge of successful manipulation places according to the robot's skills. An important aspect of Generalized Success Models is that the robot is treated as a black box, so that algorithms and control routines that are running on the robot are compiled into the Generalized Success Model.

Online, successful manipulation places are computed by instantiating Generalized Success Models and the result is represented in an Action-Related Place. Given a table and an object that is positioned on the table, an Action-Related Place is the set of pairs $\langle pos, p \rangle$, where pos is the robot's base position relative to the table, and p is the predicted success probability for grasping the object from base position pos. By maximizing p, optimal manipulation places can be found. Therefore, Action-Related Places represent places from where the current manipulation task is estimated to be successful. From a bayesian standpoint ARPLACE probability distributions are closely related to the way of human decision making.

1.2.2 Learning a Model of Places for Successful Manipulation

Generalized Success Models are learned through experience-based learning, just as humans develop internal models. In simulation, the robot navigates to many different base positions and tries to grasp target objects from there. Support vector machines are used to generalize over the training data and point distribution models are used to learn a compact, yet precise model of successful manipulation places. The approach of experience-based learning is able to capture robot skills that are hard to discover and represent in analytical modeling. It is less error-prone, too.

There are several advantages of using point distribution models as the basis for Generalized Success Models. First, point distribution models are very compact with approximately 1800 bytes. Second, they can be queried very quickly with just 40ms on a current laptop computer. Fast computation time is critical in order to enable least commitment planning. Third, it is possible to take state estimation uncertainties of the robot into account.

1.2.3 Online-Reasoning with Action-Related Places

Having explicit representations of ARPLACEs does not only tell the robot where to go in order to pick up an object. It also enables other inferences. For example, it tells the robot the *optimal* manipulation place, which is the global maximum of the ARPLACE probability distribution. Another application is to determine whether the current position of the robot is good enough or repositioning is necessary. In situations where the robot has to consider additional task constraints pertaining to execution time or energy consumption, these task goals are unified to utility values.

Usually, a robot has no ground truth data and must explore its environment with its perception systems. Sensor information, however, is noisy which leads to uncertainty of the perception systems into the world's state. ARPLACEs are able to take uncertainty of the vision system into the object's pose and uncertainty of the localization system into the robot's base position into account. This allows for more robust manipulation places when state estimation uncertainty is high.

Although sensor data is noisy, state estimation uncertainty tends to decrease as the robot moves closer or new sensor data comes in. Action-Related Places can be updated almost instantaneously and are therefore able to take incremental sensor information into account. This enables a robot to start moving with a good enough guess about promising manipulation places without committing to a particular place. Thus, the decision about the final manipulation place can be delayed until maximal sensor information is available.

Planning is also supported by the ARPLACE framework. Transformational planning can be used to optimize places when multiple manipulation tasks can be performed at once. ARPLACEs do not only enable the robot to reason about multiple manipulation tasks but also to reason about sequences of actions. The robot is able to compute a place where an object can be picked up and put down at the same time.

Moreover, ARPLACEs are a valuable source of information for high-level planning systems. In case grasp success probability is low and state estimation uncertainty is high, additional exploration may lead to a more exact determination of manipulation places. If grasp success probability is low and state estimation uncertainty is low, then the task seems to be challenging and the robot may be better off by either aborting task execution or asking a human for help.

1.2.4 Evaluations and Publications

The validity of the above claims are evaluated in the Gazebo simulator. It is shown that moving to a base position that is proposed by the framework of Action-Related Places results in manipulation places that are superior to other place-finding strategies in a statistically significant manner. Moreover, it is shown that the statistical significance of superiority increases as state estimation uncertainty increases. Applying transformational planning reduced the average execution time from 48 to 32 seconds when multiple objects had to be grasped at once.

Action-Related Places were implemented on the B21r real robot and evaluated as described by Beetz et al. (2010). However, it was not possible to prove superiority of manipulation places in a statistically significant manner. The reason is that the performance of Action-Related Places can not be measured in isolation, but has to be measured with respect to the overall performance of the robot in manipulation tasks where many systems such as localization, navigation, arm control, and computer vision interact. While it is possible to implement robot systems that robustly perform mobile manipulation in simulation, this task is not yet solved in the real world.

A case study provides a hands-on presentation of how ARPlaces are computed in complex scenarios. It further gives an intuition of how ARPlaces change under varying environmental settings. Research on Action-Related Places resulted in several peer-reviewed publications (Beetz et al., 2010; Stulp et al., 2009b,a; Fedrizzi et al., 2009; Stulp et al., 2009c).

Figure 1.3 depicts three robots that are used in our lab. All three are mobile manipulation platforms whose manipulation skills can significantly benefit from the framework of Action-Related Places. The work within this thesis was performed on the B21r robot.

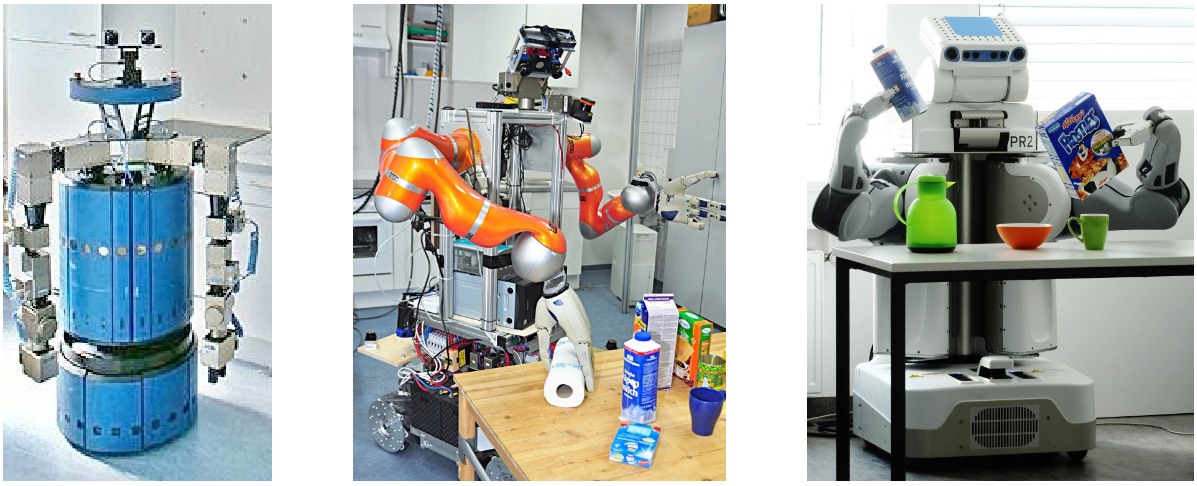


Figure 1.3 Left: B21r mobile robot with Powercube arms. Center: TUM-Rosie. A Kuka-based mobile manipulation platform. Right: PR2 manipulating milk and Frosties.

1.3 Outline of the Thesis

The remainder of this dissertation thesis is structured as follows.

Chapter 2. Preliminaries: Lays the groundwork for the framework of Action-Related Places. A manipulation scenario where a robot has to clean a table is introduced in section 2.1. Several pitfalls for computing optimal manipulation places and how these pitfalls are addressed by the ARPLACE framework are described in an informal way. Section 2.2 describes interfaces of the ARPLACE framework to other robot subsystems and the interplay of ARPLACE components. Section 2.3 discusses related work.

Chapter 3. Learning Generalized Success Models: Presents Generalized Success Models which are a precise, yet compact model of promising manipulation places. An introduction in section 3.1 specifies the problem statement and discusses related work. We describe how training data is gathered in section 3.2, how support vector machines transform the data into classification boundaries in section 3.3, and how point distribution models generalize over classification boundaries in section 3.4. Section 3.5 addresses the question of how to speed up the time for gathering training data, and section 3.6 explains how we can generalize the learning process so that Generalized Success Models that were learned for grasping a certain object can be applied to grasping other objects.

Chapter 4. Action-Related Places: After describing related work in section 4.1 we show how Generalized Success Models are used to compute Action-Related Places in section 4.2. An important part of this section is how the robot accounts for state estimation uncertainties that arise from its localization and vision system. A performance analysis evaluates how fast ARPLACEs can be computed. We vary all task-relevant parameters in section 4.3 and compute corresponding ARPLACEs in order to illustrate how manipulation places change for different manipulation tasks. In section 4.4 we explain that ARPLACEs provide valuable information for high-level planning systems and how this information can be used. Section 4.5 shows that manipulation places that are proposed by the ARPLACE framework lead to a higher probability of successful manipulation than places that are proposed by an optimal ad-hoc approach.

Chapter 5. Refining of Action-Related Places: Presents methods that improves the generality of Action-Related Places towards many directions. Section 5.1 discusses related work. Section 5.2 describes how ARPLACEs are computed for grasping objects at different grasp points and from different approach directions. Section 5.3 introduces multi-modal ARPLACEs that can emerge when the target object is reachable from multiple table edges. How ARPLACEs are computed for grasping with the left arm when there is only a Generalized Success Model for grasping with the right arm is explained in section 5.4. Section 5.5 shows how to take obstacles into account, and 5.6 presents how to grasp multiple objects at once. How uncertainty into the target object's type is taken into account is described in 5.7, and 5.8 introduces the repeller probability distribution for integrating unexpected experience such as failing to grasp an object when the ARPLACE predicted the grasp to succeed with high probability.

Chapter 6. Utility Framework for Action-Related Places: Presents a utility framework that enables the ARPLACE framework to be applied to a broad range of scenarios with many, potentially conflicting task goals. Section 6.1 describes related work, and section 6.2 introduces the utility framework. The utility heuristic is created online by a high-level planner in order to precisely reflect the importance of different constraints for the task at hand. An evaluation in section 6.3 concludes the chapter.

Chapter 7 Case Study: Presents a hands-on analysis of the ARPLACE framework in a complex scenario. The scenario is introduced in section 7.1, and section 7.2 describes the robot's actions until it finds the first target object. Section 7.3 describes how a manipulation place for the particular situation is computed, and section 7.4 shows how the ARPLACE is updated as new sensor data comes in. How the ARPLACE changes when the robot detects a second object is analyzed in section 7.5, and how the robot reacts when suddenly detecting an obstacle that blocks the most promising manipulation places is described in section 7.6.

Chapter 8 Conclusion and Future Directions: We conclude with a summary of this dissertation thesis and a discussion of future work.

Chapter 2

Preliminaries

This chapter lays the groundwork for the framework of Action-Related Places. In section 2.1 we introduce a manipulation scenario where a robot has to clean a kitchen table and the problems that arise during task execution. We further give an intuition of how Action-Related Places can help to solve these problems. We proceed by giving an overview of the system architecture of the ARPLACE framework in section 2.2. Related work is discussed in section 2.3.

2.1 Manipulation Scenario

Imagine a scenario as depicted in the left image of Figure 2.1. A mobile robot has to "clean the kitchen table" after its owners finished breakfast. We see that there are two target objects on the kitchen table: a glass and a cup.

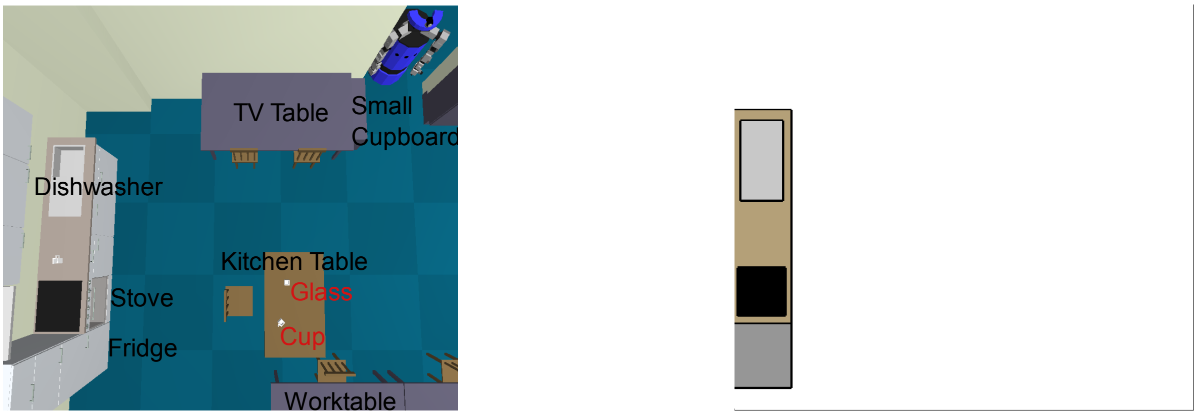


FIGURE 2.1 Left: Scenario before robot enters the kitchen. Right: The robot's internal representation of the scenario according to its current knowledge.

The robot knows the position of the dish washer, because the robot remembers the dishwasher to be next to the stove and the fridge, and the position of all these objects remained static in the past. The robot however is not completely certain where the kitchen table is located. Although the robot has a good guess, it experienced that the kitchen table's position changes several centimeters from time to time. Additionally, the robot does not know how many and which objects are located on the kitchen table. The right image of Figure 2.1 shows the robot's internal representation of the scenario according to its current knowledge.

The robot uses his knowledge base to infer that "cleaning the kitchen table" means to pick up all objects that are located on the table and put them into the dishwasher. The robot divides the task of "clean the kitchen table" into a plan with several subtasks.

1. Find the kitchen table
2. Find target objects on the kitchen table
3. Grasp target objects and put them into the dishwasher

As the robot enters the kitchen, new sensor data arrives. The robot is able to detect several objects, including the kitchen table and an object (obj_1) that is located on it. The vision system predicts obj_1 to be a glass, because the detected shape matches with cups and glasses. But because no handle is detected, the probability of obj_1 being a glass is higher. The updated internal state of the robot is depicted in the left image of Figure 2.2. Regarding the above plan, the robot now has to pick up the target object and put it into the dishwasher. But what is a good base position for grasping? Especially when taking into account that the robot prefers to grasp glasses by approaching them with the gripper coming from the top. Is it more promising to use the left or the right arm? A naive approach would consider the arm length of the robot and propose any base position from where the target object is within reach. Such base positions are colored green in the left image of Figure 2.2. This raises another issue. From which table edge should the grasp be performed?

The boolean estimation of promising base positions is very coarse. Manipulation might not succeed from all base positions within green regions, although it is kinematically possible. This can have several reasons. Motion planning systems might not find satisfying reaching trajectories even if there is one. Controllers might wind up in singularities or local minima while executing a reaching trajectory. Perception systems might not be able to satisfyingly locate the target object, or the robot moves to a base position which he believes is in a red region but because of localization uncertainties the robot's real base position is not.

The framework of Action-Related Places is able to provide a more detailed analysis of promising manipulation places that is based on grasp success probability. Action-Related Places explicitly take the robot's skills into account such as manipulator kinematics, the controllers it uses, the parameterization of these controllers, as well as uncertainties that arise

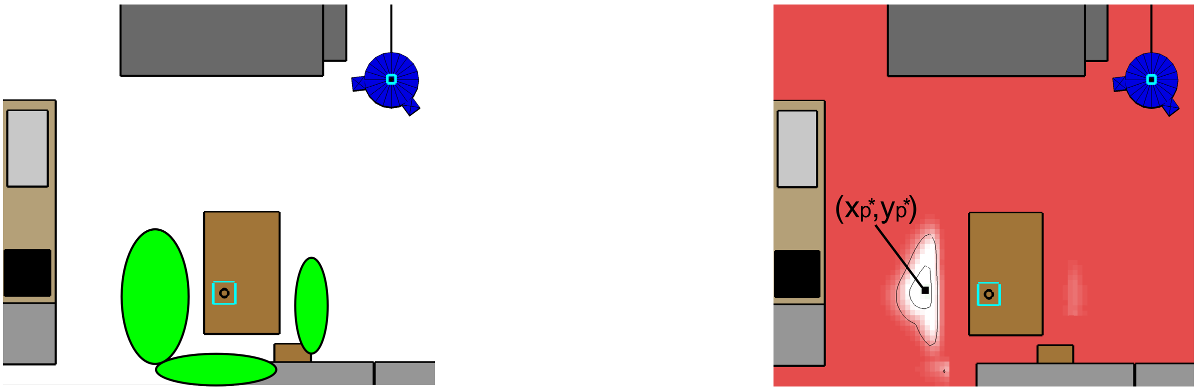


Figure 2.2 Left: Robot enters kitchen and discovers the kitchen table and an object on top of it. The cyan squares visualize the robot's uncertainty into its base position (cyan square in robot) and into the target object's pose (cyan square around object). The bigger the square, the higher the uncertainty. Green regions mark an ad-hoc approximation of the area from where the object is probably within reach. Right: Grasp success probability for grasping the object. (x_{p^*}, y_{p^*}) is the grid cell with maximum grasp success probability. Isobars are plotted at levels of 20% and 50% grasp success probability.

during state estimation. This is achieved by learning a precise model of successful manipulation places called Generalized Success Model.

Generalized Success Models are used online to compute Action-Related Places (ARPlaces). Space is discretized into grid cells and for every grid cell the Generalized Success Model is used to estimate the probability that the robot is able to successfully perform the manipulation action when it is executed from within this grid cell. The right image of Figure 2.2 depicts an ARPlace probability distribution. (x_{p^*}, y_{p^*}) represents the grid cell that maximizes grasp success probability. Isobars would be plotted at levels of 20%, 50%, and 80% grasp success probability. Considering that only the 20% and 50% isobars are drawn indicates that in this case the most promising base positions have a grasp success probability of more than 50% and less than 80%. Please note that it is not the fault of the ARPlace framework that there are no manipulation places with higher grasp success probability. The problem is that the robot is far away from the target object, making it hard to predict its pose. However, if the object's pose is highly uncertain it is impossible to be sure about good manipulation places, because the object's true pose might be far away from the estimated one. Therefore, the ARPlace framework computes optimal manipulation places for given state estimations, but maximum grasp success probability is limited by state estimation quality.

Although grasp success probability in our example is not particularly high, it is clear that the robot should perform the grasping action from the left table side. This is enough information to start moving. While moving towards the left table side, new sensor data comes in and enable the vision system to estimate the object's pose with higher accuracy. This can be seen in the left image of Figure 2.3, where lower state estimation uncertainty is reflected by a smaller cyan square around the cup. As a result the ARPLACE probability distribution gets more certain about good manipulation places and maximum grasp success probability p^* approaches 100%.

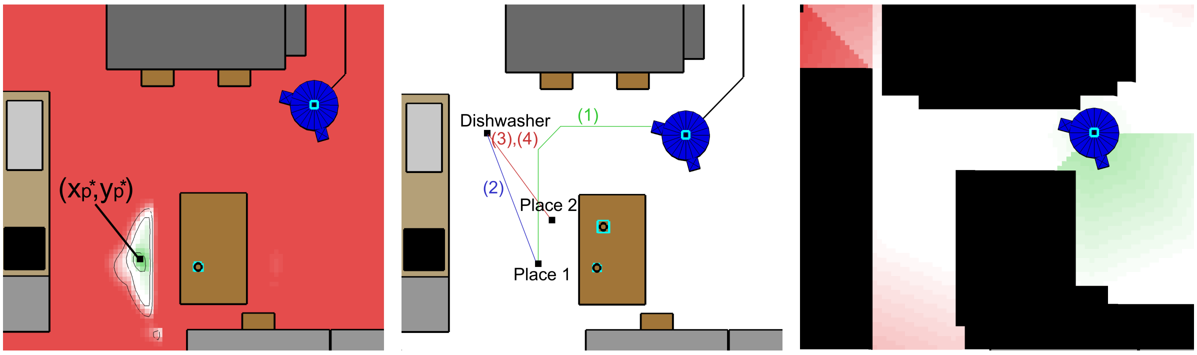


FIGURE 2.3 Left: ARPLACEs get updated as new sensor data arrives. Center: The robot's navigation trajectory in order to execute plan A. Each path is annotated with the corresponding plan step in brackets and the estimated travel time for this path segment. Right: Estimated travel time from the robot's current base position to every grid cell. Green indicates low travel time, white and red successively longer times. Unreachable cells are black.

Instead of committing to a specific base position in advance, the ARPLACE framework enables least-commitment planning because a whole range of base positions are predicted to be successful or at least probable. A least-commitment implementation enables the robot to start acting with a good enough guess, while delaying the final decision until it has to be made. This assures that maximum sensor information is available when actually committing to a goal position. The principle of least commitment is especially powerful in real environments, where complete information that is required to compute optimal goal positions is not available. Even if the environment is completely observable, dynamic properties can make an optimal, pre-planned base position suboptimal or inaccessible.

As the robot continues moving, it recognizes a second object (obj_2) on the kitchen table. This introduces new possibilities of cleaning the table. The following plans all achieve the goal of "clean the kitchen table".

A) Grasp obj_1 | Put obj_1 into dishwasher | Grasp obj_2 | Put obj_2 into dishwasher.
B) Grasp obj_2 | Put obj_2 into dishwasher | Grasp obj_1 | Put obj_1 into dishwasher.
C) Grasp obj_1 | Grasp obj_2 with other arm | Put obj_1 and obj_2 into dishwasher.
D) Grasp obj_2 | Grasp obj_1 with other arm | Put obj_1 and obj_2 into dishwasher.
E) Grasp obj_1 and obj_2 from same position | Put obj_1 and obj_2 into dishwasher.

The robot's navigation trajectory in order to perform plan A) is depicted in the center image of Figure 2.3. Finding the optimal plan is not straightforward. When only taking grasp success probability of the next manipulation action into account, then the robot will never perform plan E). The reason is that grasping just one object, but from the perfect place always results in higher grasp success probability. Plan E) leads to faster task execution, but at the drawback of moving to a base position that is not optimally for grasping either obj_1 or obj_2. In order to find the optimal plan for the current situation, the robot has to know about the task context and trade task constraints such as successful grasping, execution time, or energy consumption. The ARPLACE framework optimizes manipulation places by using decision theory and elaborating the concept of grasp success probability to grasp utility. The right image of Figure 2.3 visualizes the robot's estimated travel time. Space is discretized into the same grid cells as for estimating grasp success probability. A motion model is used for estimating travel time from the robot's current base position to each grid cell. The grasp utility of a grid cell is computed by merging its estimated grasp success probability and estimated travel time.

Figure 2.4 depicts utility distributions for grasping obj_1 (left plot), grasping obj_2 (center plot), and grasping both objects at once (right plot). To distinguish ARPLACEs that are based on grasp success probability and ARPLACEs that are based on grasp utility, grid cells with high grasp utility are colored blue instead of green. It can be seen that the best manipulation places for grasping both objects at once are between the objects. Furthermore, the ARPLACE framework found out that when grasping both objects at once, then obj_1 should be grasped with the right arm and obj_2 should be grasped with the left arm.

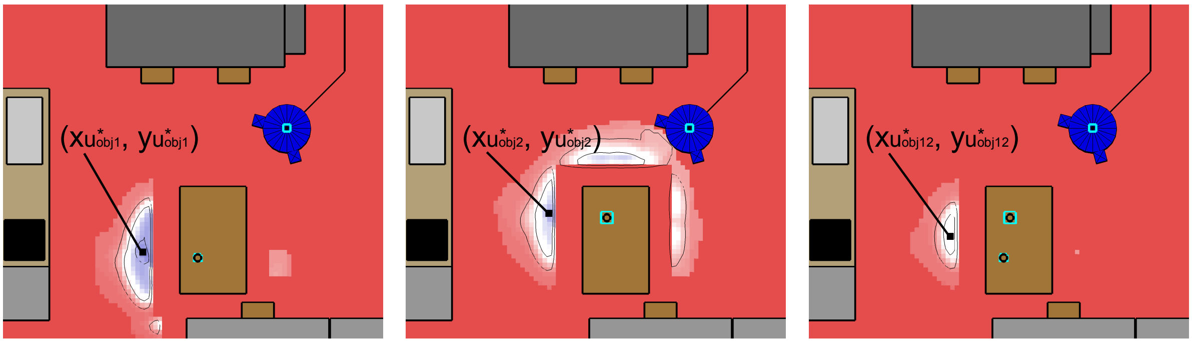


FIGURE 2.4 Left: ARPLACE utility distribution for grasping obj_1. $(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$ is the grid cell with maximum grasp utility for grasping obj_1. Center: ARPLACE utility distribution for grasping obj_2. Right: ARPLACE utility distribution for grasping obj_1 with the right hand and obj_2 with the left hand. Blue indicates high grasp utility, white and red successively lower utility.

The plots in Figure 2.4 depict grasp utility for the next action. In this case, moving to $(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$ and grasping obj_1 from there is the optimal solution because it has the highest utility among all three utility distributions which is indicated by the dark blue color and the isobars in the plots. However, when computing overall plan utility considering all plan steps, then plan E is optimal. Mainly because execution time is significantly lower. Plan A and plan B, for example, require twice as much time. That is why the new goal position is set to $(x_{u^*_{obj_{12}}}, y_{u^*_{obj_{12}}})$ for grasping obj_1 and obj_2 at once..

As the robot moves around the table, it discovers a chair that was previously hidden. Unfortunately, the chair blocks the current goal position. An open-loop system that chooses the goal position in advance may bump into the chair. Least commitment planning enables the robot to refine its decision. The updated ARPLACE utility distributions are depicted in Figure 2.5. The utility distributions show that grasping both objects at once is not promising any more. The robot decides to execute plan D): start with grasping obj_2 from base position $(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$, then grasp obj_1 from base position $(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$, and then bring both objects to the dishwasher.

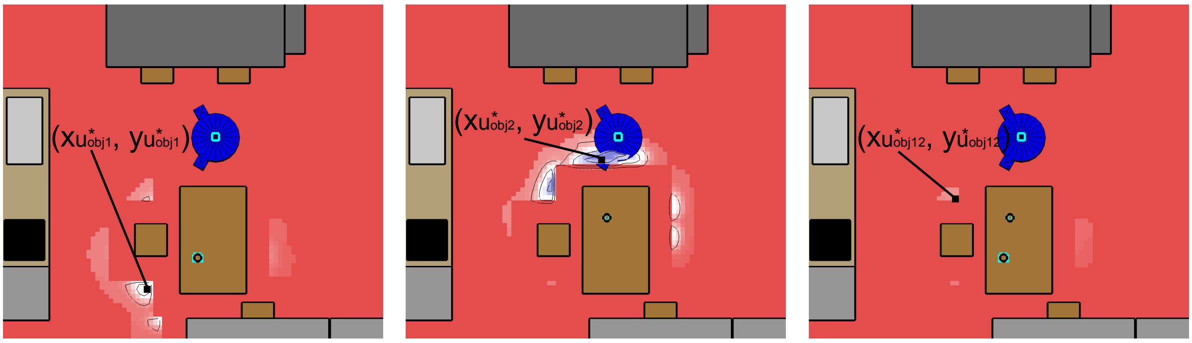


FIGURE 2.5 ARPLACE utility distributions after the robot discovered the chair. Utility distribution for grasping obj_1 (left plot), grasping obj_2 (center plot), and grasping both objects at once (right plot).

After the robot grasped obj_2 it moves towards $(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$. After finishing the last turning motion, the vision system detects that obj_1 has a handle, which was previously hidden by the object's body. The robot realizes that obj_1 is a cup and not a glass. Cups however are not grasped like glasses. Glasses are grasped by approaching them from the top, while cups are grasped at their handle by approaching them from the side. Base positioning also depends on the grasping motion the robot executes. Figure 2.6 depicts ARPLACE utility distributions before the robot detected the handle on the left image, and after the robot detected the handle on the right image. It can be seen that the ARPLACE for grasping from the side is smaller and has lower maximum utility. The reason is that grasping from the side puts additional kinematic constraints on the manipulator, like endeffector orientation.

This concludes a short glimpse on how the ARPLACE framework finds optimal manipula-

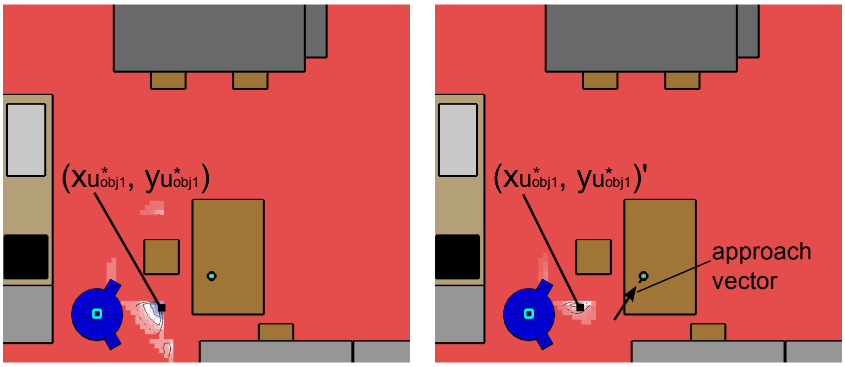


FIGURE 2.6 Left: Utility distribution for grasping a glass. Right: Utility distribution after the robot discovered the handle and realized that obj_1 is a cup that has to be grasped from the side. The arrow visualizes from which direction the gripper has to approach the handle.

tion places. We will revisite the scenario in chapter 7 and examine it in greater detail.

2.2 System Overview

Figure 2.7 depicts how the ARPLACE framework is embedded within a robot system. The ARPLACE framework receives data from three subsystems. The *localization system* provides estimations about the robot's base position, which is a 2D pose $\langle x_{\text{rob}}, y_{\text{rob}}, \psi_{\text{rob}}\rangle$ that represents the robot's mean base position and angular orientation with respect to the world frame. The localization system further provides a 3×3 covariance matrix that reflects the uncertainty of the localization system into the estimated mean base position. The *vision system* provides estimations about poses of target objects. Each pose estimation is represented by a 3D pose $\langle x_{\text{obj}}, y_{\text{obj}}, z_{\text{obj}}, \phi_{\text{obj}}, \theta_{\text{obj}}, \psi_{\text{obj}}\rangle$ that represents the object's mean pose and orientation with respect to the world frame. A 6×6 covariance matrix reflects the uncertainty of the vision system into the estimated mean pose. A *high-level planning system* is aware of the task context and knows whether it is mandatory to perform a task as quickly as possible, or whether manipulation quality is the primary goal. If new sensor data arrives, all mentioned subsystems immediately update their state estimation and forward it to the ARPLACE framework.

The online part of the ARPLACE framework iteratively receives the above mentioned input parameters and uses Generalized Success Models in oder to compute ARPLACEs. The optimal manipulation place is the one that maximizes grasp success probability or grasp utility within the ARPLACE distribution. The optimal manipulation place is forwarded to the *navigation system* as the current goal position. The ARPLACE distribution is further transmitted to a *transformational planning system* and the *high-level planner*. The transformational planning system is able to further analyse the ARPLACE distribution. For example, it can decide whether it is preferable to move towards a base position that enables the robot to grasp multiple objects at once. If this is the case, then the transformational planning system transmits the new goal pose to the navigation system and overwrites the place that was proposed by the ARPLACE framework. The high-level planning system uses the ARPLACE distribution as a source of information for being able to reason about manipulation tasks. If, for example, the ARPLACE distribution does not contain promising manipulation places, this can have several reasons. If the robot is currently uncertain about its base position or about the target object's pose, then additional exploration might lead to an improved understanding of the environment, and hence better manipulation places will be found. However, if the robot is already certain about its environment, then the task is probably difficult and the robot may be better off by either aborting the task, or asking a human person for help.

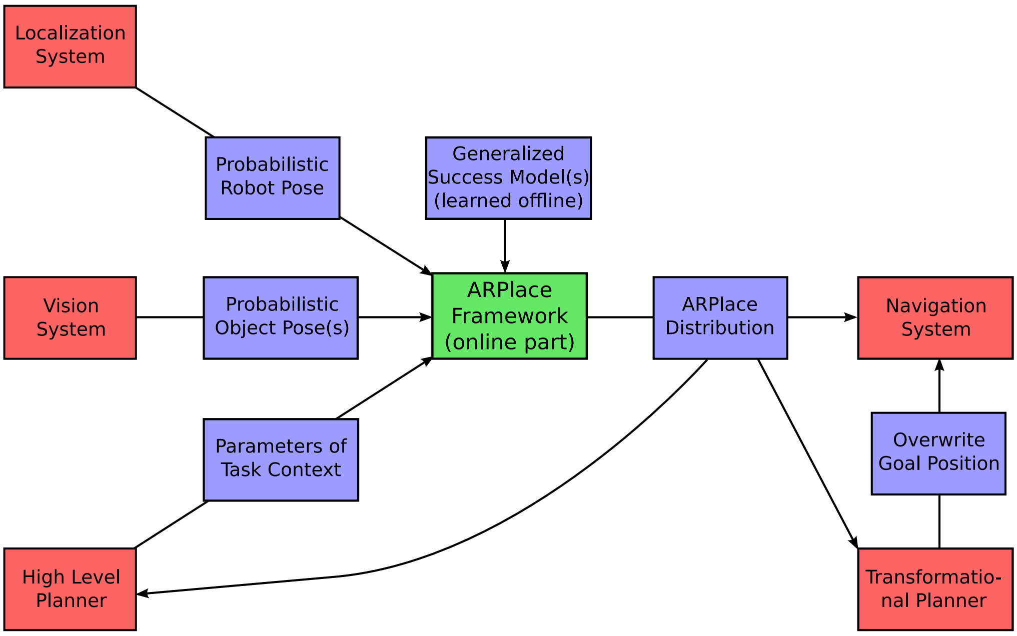

Localization System
Probabilistic Robot Pose
Generalized Success Model(s) (learned offline)
Vision System
Probabilistic Object Pose(s)
ARPlace Framework (online part)
ARPlace Distribution
Navigation System
Parameters of Task Context
Overwrite Goal Position
High Level Planner
Transformatio-nal Planner

FIGURE 2.7 Overview of how the ARPLACE framework is embedded in a robot system. Red rectangles refer to robot subsystems. The ARPLACE framework is a subsystem as well, but colored green to visualize that it is in the center of interest. Blue rectangles refer to data that is transmitted from one subsystem to another. Arrows visualize the direction of information flow.

An overview of the ARPLACE framework is depicted in Figure 2.8. It can be divided into an offline and an online part. The offline part develops models of promising manipulation places for grasping different types of objects. This is done by executing navigate-reach-grasp action sequences in simulation and analyzing the resulting data through experience-based learning. The learned model is called Generalized Success Model, which is a Point Distribution Model that is a precise, yet compact model of successful base positions. The approach of experience-based learning ensures that the robot's skills, such as manipulator kinematics, motion controllers, and the parameterization of these motion controllers are implicitly compiled into the Generalized Success Model. Further details are provided in chapter 3.

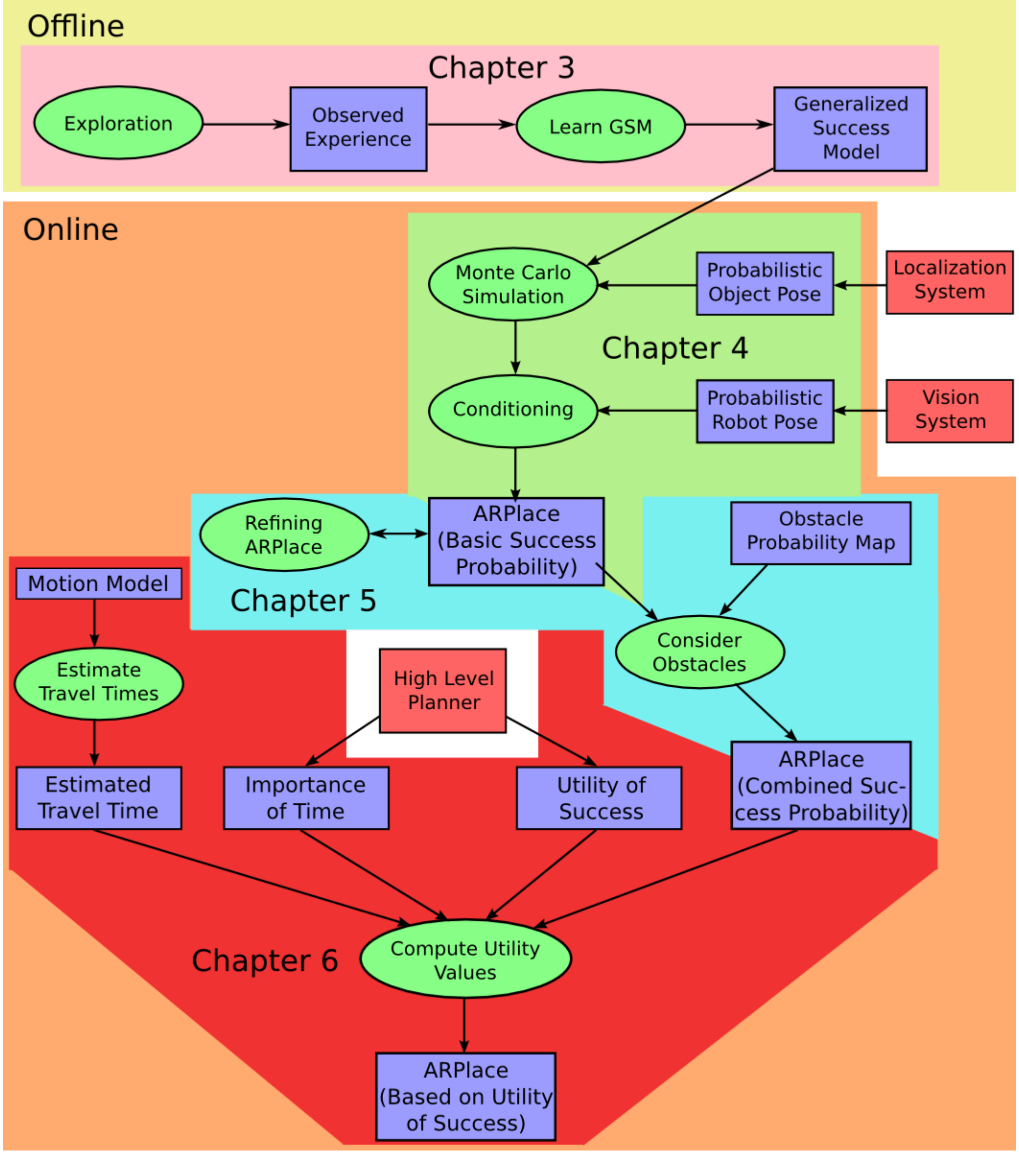


FIGURE 2.8 Overview of the ARPLACE framework. The overview depicts computational components of the offline part (yellow area) and the online part (orange area), and how they are related to each other. Components in the pink area are described in chapter 3, components in the green area are described in chapter 4, and components in the blue and red area are described in chapters 5 and 6. Red rectangles indicate subsystem that do not belong to the ARPLACE framework, but provide required information.

The online part is for finding the optimal base position from where the manipulation action can be performed as easily as possible. The robot probabilistically estimates its base position and the poses of target objects. A Monte Carlo simulation approach is used to sample object poses, and to query the Generalized Success Model. The result is a probabilistic representation of promising manipulation places. The next step is to take the robot's uncertainty into its base position into account by performing probabilistic conditioning. The result is an ARPLACE that maps base positions to the probability that the subsequent manipulation action will succeed, when it is performed from this base position. We call such Action-Related Places ARPLACEs that are based on (basic) grasp success probability. Further information on how ARPLACEs are computed online by querying the Generalized Success Model and considering state estimation uncertainties is described in chapter 4.

There are many examples, where one ARPLACE is not enough. When grasping a single object, it has to be decided which arm to use. When there are several target objects, the question emerges if multiple objects should be grasped at once. If this is the case, then it has to be decided which object to grasp with which arm. In case objects can be grasped from multiple table edges, a multimodal ARPLACE emerges. When the robot is uncertain about the type of an object, multiple ARPLACEs have to be computed in order to account for the different types of grasps that may be required. In all these examples, the ARPLACE has to be refined by computing multiple ARPLACE probability distributions and merging them into a new ARPLACE probability distribution that represents all these aspects. A special type of ARPLACE refinement is to consider obstacles. While the above operations are optional in order to find better manipulation places, taking obstacles into account is mandatory to avoid the robot being damaged. The ARPLACE probability distribution that results after taking all additional aspects into account is said to be based on (combined) grasp success probability. Refining Action-Related Places is explained in chapter 5.

However, grasp success may not be the robot's only concern. In the presence of humans, the robot should prefer to stay within the humans' field of view. If a task is urgent, performing the task as quickly as possible has priority. A utility framework generalizes the framework of Action-Related Places to take all, potentially conflicting task goals into account, and makes it applicable to a broader range of tasks and goals. The utilities we consider here are travel time and utility of successful grasping. Travel time is estimated by applying classical search techniques in combination with a motion model that is tailored towards the robot's navigation system. The importance of saving time and utility of success are evaluated by a high-level planning system according to the robot's current believe state, current task constraints, and the current task context. Together with the ARPLACE based on combined success probability the ARPLACE framework computes a utility-based ARPLACE. The base position with maximum utility is proposed as the goal position for navigation. The utility framework for

Action-Related Places is explained in 6.

The above computations are repeated when the localization system, vision system, or high-level planner provide new data. Instead of committing to a manipulation place in advance, the ARPLACE framework iteratively refines ARPLACE distributions, leading to more elaborate suggestions of promising manipulation places.

2.3 Related Work

There are many research fields that partly address the mobile manipulation problem such as manipulator kinematics and dynamics, motion and manipulation planning, as well as control of mobile manipulators. What makes mobile manipulation particularly challenging is that fields that seem to be unrelated to mobile manipulation must be considered as well. Perception and computer vision are necessary for a robot to find target objects, locomotion, localization and mapping is required so that the robot can move to a position where this object is within reach, and AI planning techniques are indispensable for obtaining a broader picture of the task in order to coordinate all subsystems. As described in the last section, the ARPLACE framework especially relies on data from the localization subsystem, vision subsystem, and the high-level planner.

To sum up, mobile manipulation is only possible if all required subsystems sturdily perform their task in a changing, partially observable environment where the robot system has to meet several constraints such as dynamic or real-time constraints. And even if all these preconditions are met a robot will not be able to successfully perform manipulation tasks if its design is poor, so that the repertoire of motions is limited by the robot's hardware.

In this section we present related work that is required to build robot systems that are able to autonomously perform manipulation tasks. A thorough overview of the field of robotics is given by Siciliano and Khatib (2008).

2.3.1 Kinematics and Dynamics

Robot kinematics studies the motion of robots, especially robot manipulators. Classic textbooks in this field were written by Craig (1986) and Sciavicco and Siciliano (2000). The foundational convention used for studying robot kinematics was introduced by Denavit and Hartenberg (1955). In a kinematic analysis, the position, velocity, and acceleration of links are calculated. Forward kinematics is used to compute the endeffector pose in 3D space when all joint angles are given, and to infer endeffector movements from joint angle movements

(Orin and Schrader, 1984). Inverse kinematics (IK) assigns an angular configuration to every joint in order to bring the endeffector to a desired goal pose as described by Pieper (1968) and Manocha and Canny (1992). Paul and Zhang (1984) showed how inverse instantanious kinematics is used for computing smooth trajectories from an initial to a goal configuration. Kinematics made it possible to develop concepts such as the dextrous workspace of a manipulator that was introduced by Waldron and Kumar (1980). Kinematics was also fundamental to find efficient manipulator designs as described by Vijakumar et al. (1986) and Tsai (1999).

The field of dynamics examines the relation between forces and motions. Featherstone and Orin (2000) applied inverse dynamics of rigid bodies to the problem of calculating actuator forces that are required to create a desired effector acceleration. This knowledge is important for designing motor controllers that are able to track desired trajectories. Knowledge in the fields of inverse kinematics and inverse dynamics were particularly important to make factory manipulators become a valuable aid in manufacturing from early hydraulically actuated Unimation platforms (Nof, 1999) to current light weight arms with advanced force-control abilities (Hirzinger et al., 2002).

In industrial manufacturing, however, the manipulator is mounted stationary in an environment that is well under control. The problem of mobile manipulation in everyday environments is more difficult, because mobile robots have to move around in an environment that is complex, dynamic, and partially observable.

2.3.2 Locomotion, Localization, and Mapping

A motion system enables mobile robots to move around and towards target objects. This makes mobile robots much more flexible than factory manipulators where the objects have to be brought to the robot. Different types of robot locomotion concepts have been developed with wheeled and legged locomotion systems being the most popular. Wheeled robots exist with different types of wheels, such as passively driven wheels, passive caster wheels, actively driven caster wheels, actively orientable wheels, swedish, or spherical wheels. Asama et al. (1995), for example, use three spherical wheels for omnidirectional locomotion. The wheels can be placed in various ways in order to ensure certain kinematic and dynamic properties such as irreducability, controllability, nonholonomy, and stability. Frequently used wheel configuration include two wheels, three wheels such as synchronous drive or omnidirectional drive robots, or car-like wheel structures with four wheels. Takahashi et al. (2001) present movements that can be obtained by driving a wheel chair robot that has two wheels. Wada and Mori (1996) describe how omnidirectional locomotion can be achieved by using three conventional wheels.

Research for legged locomotion focussed on achieveing static and dynamic stability. The analysis of cyclic walking led to robots that were able to walk passively. This line of research was pioneered by McGeer (1990). Especially the zero-moment point (ZMP) is considered to be an important result in robotics research. Vukobratovic and Borovac (2004) presents a comprehensive overview about the zero-moment point. Controllers that are based on forward dynamics and the zero-moment point enabled biped robots such as Johnnie (Gienger et al., 2001) and ASIMO (Masato and Tooru, 2001) to walk.

In order to keep track of its position, a mobile robot has to build models of its environment, usually referred to as maps. Moravec and Elfes (1985) were among the first that created maps from sensor data. They used a sonar sensor as information source. The problems of localization and mapping are related to each other. It is difficult for a robot to localize itself without a map, and it is difficult to build a map when the robot does not know the position of its base. Therefore, techniques that perform synchronous localization and mapping (SLAM) have been proposed. Moutarlier and Chatila (1989) adressed the SLAM problem by implementing an algorithm that is based on Kalman filtering. Leonard and Durrant-Whyte (1991) tried to reduce the uncertainty that arises because of imprecise and erroneous sonar information. They use multiple sonar sensors that identify landmarks and continuously track them.

Today's state of the art localization algorithms are probabilistic. Fox et al. (1999) introduced Monte-Carlo localization based on particle filters are considered to be especially well suited for representing and maintaining a probability distribution about the robot's position with high accuracy and acceptable computational cost. Laser range scanners such as triangulation sensors, phase-modulation sensors, and time-of-flight sensors are currently the dominant type of sensor in order to acquire data for localization and mapping. Blais (2004) presents a thorough overview of 20 years of laser range sensor development from single point laser scanners to time-of-flight systems.

2.3.3 Planning in AI

Generalized Success Models are a form of action models, and concepts such as ARPLACE probability distributions are typically represented in AI planning as components of action models, such as preconditions that must be verified by the planning algorithm to ensure that an action will succeed and have their specified effects. In the context of Action-Related Places, the precondition for a grasping action would be that the robot has moved its base to a position where grasp success probability is high. Action models are often stated in a variant of the planning competition language PDDL. Fox and Long (2003) extended PDDL in order to improve planning in temporal domains. Younes (2003) extended PDDL to include probabilistic models and to restrict forms of concurrent and continuous effects. Approaches to make action models more realistic were described by Schmill et al. (2000). They allow to express the effects of a robot's action in a dynamic, partially-observable environment. Operator models were learned by getting data from simple interactions between an agent and its environment, such as moving and turning the base or lifting and lowering the gripper. Clustering and decision tree induction was subsequently applied on the acquired data. Another approach that was presented by Gravot et al. (2003) and Cambon et al. (2004a) proposes to ground action preconditions like *being within reach* in the existence of a motion plan.

While the representations listed above are almost exclusively designed for the computation of action plans, action-related concepts such as Action-Related Places are designed to enable much broader reasoning functionality. For example, ARPLACEs equip the robot with predictive decision making capabilities. Wolpert and Kawato (1998) and Flanagan et al. (2003) have shown that prediction-based control is a powerful concept in the context of human and animal motion.

An interesting line of research that shares paradigms with action-related concepts like learning the relation between objects and actions, or building prediction models are Object-Action Complexes (OACs). Geib et al. (2006) and Wörgötter et al. (2009) present OACs that can be used to integrate high-level artificial intelligence planning technology and continuous low-level robot control. The work stresses that objects and actions are inseparably intertwined and should therefore be paired in a single interface. By physically interacting with the world and applying machine learning techniques, OACs allow to acquire high-level action representations from low-level control representations. OACs are meant to generalize the principle of affordances that was introduced by Gibson (1977). Affordances represent the relation between a situation usually including an object of a defined type, and the actions that it allows.

2.3.4 Perception

Objects are constantly changing their position in everyday environments, mostly because of pick and place actions of human people. Mobile robots that perform manipulation tasks need techniques for finding required objects. Computer vision plays a crucial role in this context. Klank et al. (2009a) describe the 3D perception system that is used at our chair. The perception system uses 2D RGB images and images from a 3D time-of-flight camera. The system is able to detect, localize, and track objects in cluttered household scenes. It is implemented within a robot system and especially designed for grasping applications. Furthermore, it provides pose estimation uncertainties via covariance matrices, which is valuable information for the ARPLACE framework. Other vision systems that tackle the task of detecting objects from images are presented by Hoover et al. (2008), Vahrenkamp et al. (2008), and Saxena et al. (2008). The last two also especially deal with object detection for performing manipulation tasks.

Although the location of household items is dynamic, it is usually not random. Televisions are found in living rooms. Knives, forks, or plates are found in the kitchen, and a bottle of milk is presumably located in the fridge. Knowledge based systems that develop common sense representations of their environment are presented by Gupta and Kochenderfer (2004) and are able to support computer vision systems by providing initial guesses on where to start searching. Semantic maps are an important way to represent this knowledge that are currently developed by Rusu et al. (2009), Nüchter and Hertzberg (2008), and Galindo et al. (2008).

2.3.5 Motion Planning

After knowing the pose of an object, the robot has to grasp it. Locomotion systems, as described above, are required to enable the robot to move close to the target object. In order not to bump into obstacles while moving, robots have to find collision-free paths. This problem is called motion planning. Lozano-Perez (1980) presents the configuration space, which is more appropriate for planning motions than 2D or 3D euclidean space.

Combinatorial approaches to motion planning were the first that have been studied. Combinatorial approaches include shortest path roadmaps which were introduced by Nilsson (1969), or cell decomposition methods as described by Schwartz et al. (1987) and Zhu and Latombe (1991). Combinatorial algorithms are exact and complete. If there is a solution to a motion planning query, the algorithm will find it. Otherwise the algorithm will correctly report that there is no solution. This requires to explore the whole state space of a problem. The state space grows according to the robot's number of degrees of freedom and additional properties of the motion planning problem such as the presence of dynamic obstacles, uncertainty about motion execution, or differential constraints. Combinatorial algorithms and their computational complexity have been analyzed thoroughly in the work of Canny (1988b). Important results are that Reif (1979) found out that the "generalized mover's problem" is PSPACE-hard, as well as several other motion planning problems with static obstacles that were published by Hopcroft et al. (1984). According to Reif and Sharir (1985) motion planning in the presence of obstacles that move with unbounded velocity modulus is NP-hard, and Canny and Reif (1987) discovered that motion planning in the presence of control uncertainty is NEXPTIME-hard. The authors "believe this to be the first instance of a provably intractable problem in robotics".

Concluding, the motion planning problem is difficult. While combinatorial planners may be applicable to problems with small state spaces such as navigation on a 2D plane, they are not tractable for larger state spaces that arise when planning motions of manipulators or entire robot systems. That is why newer research published by Barraquand and Latombe (1990) and Brooks and Lozano-Perez (1985) abandoned the property of completeness and proposed sampling-based approaches. Sampling-based approaches do not search the whole configuration space but construct a search tree where each node represents a robot configuration. The robot's current state and the robot's goal state are initially inserted into the search graph and the algorithm tries to connect them. New robot configurations are sampled and inserted into the search tree, if the new state is collision free and can be connected to an existing node. Yershova et al. (2005) and LaValle (2006, chap. 5) present sampling techniques that are mainly tailored towards achieving fast exploration of the state space. Yershova and LaValle (2007) describes how to efficiently find the nearest neighbor for a sampled configuration. Rapidly Exploring Random Tree algorithms (RRTs), as presented by LaValle (1998), are sampling-based planners which are considered to be the current state of the art in motion planning. The framework of RRT-based motion planning was significantly improved by LaValle and Kuffner (2001). RRTs incrementally construct a search tree that gradually improves resolution until the tree densely covers the space, as described by Lindemann and LaValle (2006). The exploration heuristic can be tailored towards motion planning problems and usually shows a Voronoi bias, which enables RRTs to quickly cover large portions of the state space. Lindemann and LaValle (2004) examines exploration strategies and the Voronoi bias of RRTs.

The solution to a motion planning problem is usually forgotten, after it has been computed and executed. When the robot can expect to face similar motion queries in the future, then it is reasonable to store parts of the constructed search tree, called roadmap, for later reuse. Canny (1988a) was among the first that proposed geometric roadmap methods, and Kavraki et al. (1996) introduced sampling-based roadmap planners. Bohlin and Kavraki (2000) applied the principle of lazy evaluation for building probabilistic roadmaps and were able to significantly reduce the number of required collision checks.

While all above methods can be applied to all motion planning problems, this may be overkill for the navigation problem due to its lower dimensionality. Borenstein and Koren (1991) developed the Vector Field histogram that is especially designed to address the navigation problem. It is based on potential fields that were investigated by Khatib (1986), and adds mechanisms for explicitly dealing with uncertain sensor information. Vector Field histograms can be used for online path planning. However, they are not complete and prone to local minima. Stilman and Kuffner (2004) and Stilman et al. (2006) considered the problem of navigation among movable obstacles from a motion planning point of view. This enables robots to move obstacles out of the way if they block paths to the goal position.

Textbooks by Latombe (1991), LaValle (2006), and Choset et al. (2005) give a detailed overview of classic and recent motion planning techniques, with the last one having a strong bias towards actual implementation on robots. Laumond (1998) focusses on non-holonomic path planning. Concluding, it can be said that motion planning is difficult and positioning the robot at a base position that simplifies the motion planning problem is a reasonable goal.

2.3.6 Manipulation and Grasp Planning

The research fields of manipulation and grasp planning try to solve the motion planning problem for specific applications. Manipulation refers to the process of moving or re-arranging objects in the environment. In order to perform manipulation actions, the robot has to establish physical contacts with a target object and subsequently move it by exerting forces and moments, as investigated by Mason and Salisbury (1985). In contrast to motion planning, where all collisions are considered harmful, manipulation planning tries to avoid collisions with obstacles but needs to find a way to position the gripper so that it collides with the target object and grasps it in a stable manner. Although many techniques from the motion planning community like sampling-based motion planning are still used, the special structure of manipulation or grasp problems is exploited to acquire faster and more robust algorithms. Alami et al. (1995) present two planners that address the manipulation planning problem by building manipulation graphs. A starting point for developing manipulation planning algorithms is OpenRAVE that was developed by Diankov and Kuffner (2008). OpenRAVE enables users to focus on planning, while providing modules to handle low-level aspects such as kinematics, dynamics, or collision checking.

Miller and Allen (1999) explores the field of grasp stability and Miller and Allen (2000) present a publicly available grasp planner that computes grasp points that lead to physically stable grasps. Fearing and Hollerbach (1985) present an approach where grasp stability is optimized by a heuristic and can be computed for arbitrary objects. Databases that contain stable grasps for many objects can help to reduce computation time. The Columbia Grasp Database (Goldfeder et al., 2009), for example, stores 238.737 grasps for 7.256 object models and several hands. Maldonado et al. (2010) describe a system for grasping objects that are not known a-priori. A vision system uses a time-of-flight range camera in order to estimate the object's center and an approximation of its shape. An algorithm for grasp pose optimization ensures that the grasp is physically stable. The problem of efficiently generating collision-free force-closure grasps for dexterous hands is adressed by Berenson and Srinivasa (2008).

When the grasp planner found an endeffector pose that is able to grasp the target object in a physically stable manner, then the next problem is to compute a manipulator configuration that brings the endeffector to the required grasping pose. Usually, the manipulator's goal configuration is computed by solving the inverse kinematics problem and the result is the goal configuration for the subsequent manipulation planning query. However, most IK queries have multiple, sometimes infinitely many goal configurations. Therefore, the approach of using inverse kinematics to compute just one goal configuration is suboptimal. It is possible that the IK solution to the inverse kinematics query is not reachable from the manipulator's initial configuration, but another IK solution is reachable, as described by Bertram et al. (2006). Berenson et al. (2009a) tackle this problem by introducing workspace goal regions. Bertram et al. (2006) avoid to compute goal configurations at all. RRT-based motion planning is performed from the manipulator's current configuration, and for every node that is attached to the RRT it is checked if it brings the manipulator closer to the goal pose by computing the node's forward kinematics. In other words, manipulation planning is done in configuration space, but the check whether the new node is closer to the goal pose is performed in workspace. Vandeweghe et al. (2007) studied a similar approach that is called Jacobian Transpose-directed Rapidly Exploring Random Trees (JT-RRT).

Further algorithms exist that apply RRTs to manipulation planning. BiSpace manipulation planning was developed by Diankov et al. (2008) and produces fast plans to complex high-dimensional problems by simultaneously exploring multiple spaces. An approach that considers uncertainty in manipulation planning is presented by Berenson et al. (2009b), where Task Space Regions are used for deciding whether the task can be achieved at all. When this is the case, then a RRT-based motion planning algorithm is used for finding a path. Vahrenkamp et al. (2009) introduce an algorithm for efficiently computing a trajectory for dual-arm manipulation and re-grasping tasks. The inverse kinematics problem is addressed by performing gradient-descent in the manipulator's pre-computed reachability space and two RRT-based algorithms are used for finding a path. The problem of planning manipulator motions in the presence of endeffector constraints is addressed by Berenson and Srinivasa (2010).

aSyMov is a roadmap-based path planner that solves complex multi-robot manipulation planning problems. It is described in the work of Cambon et al. (2003), Cambon et al. (2004a), and Cambon et al. (2004b). aSyMov explicitly takes geometric constraints into account and uses symbolic knowledge in order to guide the search. It is worth mentioning, that aSyMov builds high-level plans of manipulation actions, where positions for grasping are represented in an abstract way.

Zacharias et al. (2007) investigate the capability map, that is a data structure that allows to quickly decide whether an object is reachable or not. Capability maps are generated by separating the workspace into discrete grid cells and trying to solve multiple inverse kinematics queries for every grid cell. Capability maps can be used to find regions that are reachable, provide a measure of the manipulator's dexterity, and can be used online to find motion trajectories by searching paths through connected grid cells that are labeled as being reachable (Zacharias et al., 2008).

2.3.7 Coupling of Navigation and Manipulation

Several papers studied the question of how to control robot manipulators that are mounted on mobile bases. Yamamoto and Yun (1999) and Tan et al. (2003) are two examples. A topic that attracted a lot of attention was the one about concurrently controlling a robot base and a manipulator. Seraji (1993) and Bayle et al. (2000) published important work within this field. Cameron et al. (1993) proposed reactive control concepts to solve this task. Brock and Khatib (2002) explored the elastic strips framework that is a recent approach of controling robot manipulators. It combines reactive control with motion planning in order to address the obstacle avoidance problem. All the aforementioned methods were primarily concerned with studying kinematics, dynamics, and control of robot manipulators and did not address the question of where to position the robot's base in order to perform manipulation actions. Seraji (1995) provides an analytic offline solution to determine appropriate base locations from which the robot can reach a target point. The concept of reachability, however, is boolean. A target object is either in reach or it is not. The approach is not able to handle state estimation uncertainties.

It is possible to find manipulation places for grasping objects that are out of reach by solving a motion planning query where the degrees of freedom of the mobile base and the manipulator degrees of freedom are considered together in a single motion planning problem. However, this leads to state spaces with high dimensionality. For example, when a robot is moving on a plane and has a manipulator with six degrees of freedom, then the corresponding motion planning problem has nine degrees of freedom. When the robot considers to use both arms for concurrent grasping, then the motion planning problem has 15 degrees of freedom.

As a result, Berenson et al. (2008) analyze pick-and-place tasks and argue that dividing them into the following subproblems is necessary in order to reduce complexity. "1) move the robot [...] to a configuration [...] near the object, 2) grasp the object, 3) move the robot [...] to some configuration which places the object into its goal configuration". The coupling between subproblems is addressed by optimizing the overall robot configuration using a metric that considers grasp quality, configuration desirability, and configuration clutter. The approach simultaneously addresses the base positioning and manipulator planning problem. Action-Related Places on the other side particularly address problem 1) of base positioning, but do this in an elaborate manner by taking additional aspects into account. State estimation uncertainty, for example, is considered. The computation of grasp success probability enables high-level planning systems to reason about the manipulation task at hand and qualitative aspects such as from which table side an object should be grasped are addressed. Moreover, Berenson et al. (2008) compute the metric for base positioning completely online, while the ARPLACE framework performs the computationally expensive part of learning a model of successful manipulation places offline.

Zacharias et al. (2009) describes a method that uses capability maps in order to position a robot for performing manipulation tasks. Diankov et al. (2008) presents a similar approach. Both approaches are different to Action-Related Places in many aspects. They start with computing a reaching trajectory based on a reachability analysis of the manipulator's workspace. The base position is subsequently chosen so that positional constraints for the computed reaching trajectory are met. Both approaches simultaneously address the base positioning and manipulator planning problem. Action-Related Places on the other side particularly address the problem of base positioning, but do this by taking additional aspects into account such as state estimation uncertainties, providing a qualitative measure of promising grasp positions by computing grasp success probability, enabling high-level planning systems to reason about the manipulation task, and are able to take multiple, potentially confliting task goals into account by using decision theory.

Pin and Culioli (2005) deal with the problem of optimizing a robot's configuration when changes occur in task requirements or task constraints. Especially load and position constraints that are applied at the end-effector are considered. But obstacle avoidance, maneuverability, and several torque functions are also taken into account. The problem of optimally positioning the robot base in order to perform a manipulation action is also treated. The resulting optimization heuristic, however, is complex.

Okada et al. (2006) call a good base placement for grasping a *spot*. Different spots are defined for different tasks, such as manipulating a faucet, a cupboard, or a trashcan. In their work, spot information is hand-coded, while the ARPLACE framework uses experience-based learning to learn a model of successful manipulation places.

Hsu et al. (1999) examine the problem of choosing a position for mounting factory manipulators in cluttered environments. Randomized path planning is used to minimize execution time for performing manipulator motions between two endeffector frames. Because factory manipulators are immobile and the environment is well under control, no issues of re-positioning the robot's base, no uncertain state estimation, no least commitment planning, and no predicitve abilities are required.

Gienger et al. (2008) optimize whole body postures for grasping an object. The goal is to obtain fluent approach and grasp motions. The approach motion specifically takes constraints on the final grasp into account and considers comfort measures on intermediate configurations. This is achieved by using object-specific task maps and combining existing techniques for grasp optimization, trajectory optimization, and attractor-based movement representation.

Ansari and Hou (1992) search an optimal base trajectory for a mobile manipulator in order to perform a sequence of tasks. A set of feasible base placements is computed and genetic algorithms are proposed for planning a path between these base placements. The approach is computationally intensive because the set of base placements is determined by exhaustive search. Obstacles are not considered, and the approach does not qualitatively evaluate how promising a base placement is.

CHAPTER 3

Learning Generalized Success Models

This chapter describes how a compact and precise representation of successful manipulation places is learned. We call this representation Generalized Success Model (GSM). In analytic modeling, the way to develop a model of successful manipulation places would be to analyze the kinematics and dynamics of the robot at hand. However, this requires extensive knowledge about the robot system. Moreover, there are many factors that are difficult to model analytically but have an impact on successful manipulation places such as the controllers the robot uses, the parameterization of these controllers, as well as uncertainties that arise during state estimation.

That is why we use *experience-based learning* for developing Generalized Success Models. A robot performs a mobile manipulation task many times from different base positions and records whether the manipulation task was executed successfully or not. This allows to learn a model of successful manipulation places without requiring knowledge of the robot system itself, because the *robot's skills are compiled* into the recorded experiment data. The robot can be considered as a *black box* that executes the manipulation task. The Generalized Success Model is then learned from the training data and is a compact and precise of the robot's skills and successful manipulation places. An important advantage of this approach is that it is not important which algorithms and control systems are actually running on the robot, because the Generalized Success Model is able to abstract them. Another benefit of Generalized Success Models is that they can be queried very fast, as we will see in chapter 4. This is a critical aspect to enable least commitment planning.

An outline of the chapter is depicted in Figure 3.1. The chapter is structured as follows. Section 3.1 presents the problem statement and discusses related work. Section 3.2 describes the process of gathering training data in simulation by performing a navigate-reach-grasp action sequence with Player and Gazebo. The B21r robot in Gazebo and exemplary data that is acquired during exploration are depicted in the first and second image at the bottom of Figure 3.1. Section 3.3 explains how support vector machines (SVMs) classify the training data in order to compute classification boundaries that capture areas of successful base positions for particular object poses. A classification boundary is depicted as a green hull in the third image at the bottom of Figure 3.1. A point distribution model (PDM) is used in section 3.4 to generalize over classification boundaries. The learned PDM is the Generalized Success Model that captures successful manipulation places for arbitrary object poses. The fourth image at the bottom of Figure 3.1 depicts several classification boundaries (colored green) and the mean of the corresponding Generalized Success Model (colored blue). Section 3.5 presents two techniques for learning an accurate Generalized Success Model with significantly less training data, and section 3.6 describes how Generalized Success Models that were learned for grasping a particular object can be used for grasping similar objects.

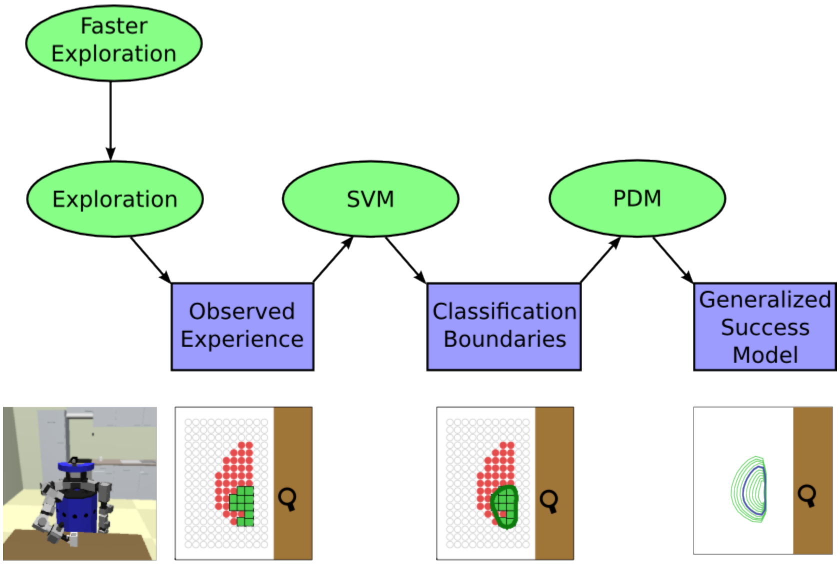


FIGURE 3.1 Computational steps for learning a GSM. Green circles represent algorithms that create and transform data. Blue rectangles represent data that is generated and passed from one algorithm to another. Images at the bottom are visualizations of data structures.

3.1 Introduction

This section provides a problem statement in 3.1.1 that explains why experience-based learning is superior to analytical modeling for developing a model of successful manipulation places. Section 3.1.2 presents related work.

3.1.1 Problem Statement

There are many factors that determine from which base positions a robot can successfully perform a manipulation action, such as the kinematics of the robot, the controllers it uses, the parameterization of these controllers, as well as uncertainties that arise during state estimation. This is why positioning the base depends on many factors, which are difficult to capture in an analytical model. Moreover, manually designing an analytical model that takes all these factors into account is tedious and error-prone. An alternative to analytical modeling is advocated in a recent roadmap paper for manipulation by Kemp et al. (2007): “it seems almost inevitable that learning will play an important role in robot manipulation”.

This is one of the reasons why we chose to learn Generalized Success Models through experience-based learning. The advantages of using experience-based learning for developing a model of successful manipulation places are as follows. Experience-based learning enables a robot to

- develop a concept of manipulation place that is tailored towards the robot’s skills such as the hardware and control programs it uses
- ground the model in real experience from interactions of the robot with the environment
- capture complex robot behavior that emerges from complex subsystems and their interplay

Appendix B provides a more thorough discussion to support the claims above. The Generalized Success Model is learned offline, and used by the online part of the ARPLACE framework in order to compute Action-Related Places. Please see Figure 2.8 for a visualization of how the Generalized Success Model relates to the overall ARPLACE framework.

3.1.2 Related Work

Using a heuristics-driven search in task space has proven to be a very effective approach to plan motion, even in complex cluttered scenes (Berenson et al., 2007). However, if everyday situations are encountered frequently, then having a set of standard solutions like skills or motor primitives for these standard situations is more effective than treating each repeated task as a novel task that requires search. Because humans use standard motion primitives so frequently, they can be optimized over time, which leads to stereotypical human motion, and improves the predictability of motions. We believe that these are desirable properties for robot behavior as well. If more complex, novel situations do happen to arise, a standard solution will not suffice, and motion planning algorithms can be used to perform search in order to find a solution. The two approaches complement each other well.

Kuipers et al. (2006) present a bootstrapping approach that enables robots to develop high level ontologies from low level sensor data including distinctive states, places, objects, and actions. These high level states are used to choose trajectory-following control laws in order to move from one distinctive state to another. Our approach is exactly the other way around: given the manipulation and navigation skills of a robot which are far too high-dimensional to learn with trajectory-following control laws, learn places from which these skills (e.g. grasping) can be executed successfully. Our focus is on action and affordance, not recognition and localization. For us, place means ’a cluster of locations from which I can execute my (grasping) skill succesfully’, whereas for Kuipers et al. (2006) it rather refers to locations that are perceptually distinct from others. Furthermore, their work has not yet considered the physical manipulation of objects, and how this relates to place.

Capability maps that were developed by Zacharias et al. (2007) are an alternative approach to modelling robot configurations that lead to successful grasping. Capability maps are generated by separating the workspace into discrete regions and trying to solve multiple inverse kinematics queries for every region. Because capability maps focus on kinematic aspects, they do not take more subtle skills of a robot into account, such as motor controllers or joint friction.

Learning success models is a form of precondition learning. Most research on learning preconditions focusses on learning symbolic predicates from symbolic examples (Clement et al., 2007). These approaches have not been applied to robots, because the used representations do not suffice to encapsulate the complex conditions that arise from robot dynamics and action parameterizations. In robotics, the focus in pre-condition learning is on grounding preconditions in robot experience. For instance, ‘Dexter’ learns sequences of manipulation skills such as searching and then grasping an object. This work is described by Hart et al. (2006). Declarative knowledge such as the length of its arm is learned from experience. Learning success models has also been done in the context of robotic soccer, for instance learning the success rate of passing (Buck and Riedmiller, 2000), or approaching the ball (Stulp and Beetz, 2008). Our methods extend these approaches by explicitly representing the region in which successful instances were observed, and computing Generalized Success Model from these regions.

Stoytchev (2009) stresses the importance of grounding the behavior of robots in observed experience. It is stated that “grounding of information based on a single act-outcome pair is not sufficient” because the result may be coincidence. Therefore, “the outcome must be replicated at least several times in the same context” which enables the robot to “build up probabilistic confidence”. Sinapov and Stoytchev (2010) consider exploration to be “one of the hallmarks of human and animal intelligence”. Applied to the domain of object recognition, the question is examined why exploratory behavior of robots are able to significantly improve recognition rates. The paper further establishes a link between empirical studies of exploratory behaviors in robotics and theoretical results on boosting in machine learning.

Support vector machines were developed within the community of statistical learning. Vapnik (1995) gives a good overview of statistical learning theory. Support vector machines are a family of algorithms for supervised learning that are used for classification and regression. A collection of early papers is published by Schölkopf et al. (1999). Classic textbooks are written by Schölkopf and Smola (2001) and Shawe-Taylor and Cristianini (2004).

Point distribution models are a technique for modeling the shape of 2D and 3D objects and were introduced by Cootes et al. (1995b). Cootes and Taylor (2004) is a more thorough and updated description. Point distribution models are used in the research community of image understanding and are especially suitable for modelling variations in medical images (Cootes et al., 1995a) and faces (Wimmer et al., 2008).

Recently, similar methods to the ones presented here have been used to determine successful grasps rather than manipulation places for grasping. For instance, Detry et al. (2009) determine a probability density function that represents the graspability of specific objects. This function is learned from samples of successful robot grasps which are biased by observed human grasps. However, this approach does not take examples of failed grasps into account. The distance between a failed and a successful grasp can be quite small and can only be determined by considering failed grasps. The classification boundaries presented in section 3.3 are similar to Workspace Goal Regions that were developed by Berenson et al. (2009a). They differ in that classification boundaries represent *base positions* that lead to successful grasping for a particular object pose, whereas Workspace Goal Regions represent *goal configurations* in a manipulation planning query.

An interesting line of research that shares some paradigms with the ARPLACE framework, such as learning the relation between objects and actions, or building prediction models are Object-Action Complexes (OACs). Geib et al. (2006) and Pastor et al. (2009) present OACs that can be used to integrate high-level artificial intelligence planning technology and continuous low-level robot control. The work stresses that, for a cognitive agent, objects and actions are inseparably intertwined and should therefore be paired in a single interface. By physically interacting with the world and applying machine learning techniques, OACs allow to acquire high-level action representations from low-level control representations. OACs are meant to generalize the principle of affordances that was introduced by Gibson (1977).

3.2 Gathering Training Data

Gathering data with a real robot is time consuming, tedious, and error prone. Current simulators have reached a decent level of maturity. However, the applicability of data gathered in simulation to the real world is highly dependent on the quality of the geometric and kinematic model of the real robot, as well as the ability of the simulator to simulate dynamic properties of the environment. Another issue that has to be achieved is that software systems that control the real and simulated robot must behave identically. We achieve this by using the robotic middleware Player (Gerkey et al., 2003), which is a language and platform independent network server for robot control and provides a consistent API for abstracting from low level hardware functionality. Gazebo is a high-quality 3D multi-robot simulator for indoor and outdoor environments. Gazebo is also Player-compatible which means that programs that are written for a simulated robot can be applied to a robot that is interacting with the real world, and vice versa. It uses the Open Dynamics Engine library (Smith, 2004) for a proper simulation of rigid body physics. A wide range of robot platforms such as a Pioneer2DX, a SegwayRMP, or our RWI B21r mobile robot can be simulated, as well as numerous sensors including laser range-finders or stereo cameras. We created an accurate Gazebo model of our B21r robot which assures that the data that is gathered in simulation is applicable for the real robot. The left and center images in Figure 3.2 depict the real and simulated robot. Appendix A gives a detailed insight into hardware and software components of the B21r mobile robot.

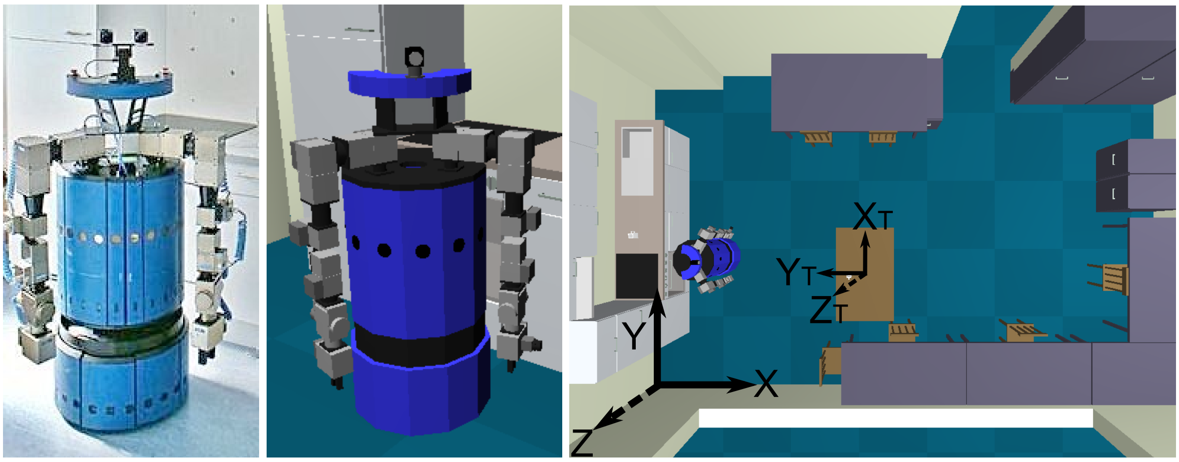


FIGURE 3.2 Left: Real B21r robot. Center: B21r robot in the Gazebo simulator. Right: Overview of the Assistive Kitchen. The coordinate frames of the world (x, y, z) and the table (x_T, y_T, z_T) are visualized. The white object on the brown table is the cup that the robot wants to grasp. The ground has a pattern of darker and brighter blue squares which measure 1m x 1m.

3.2.1 Experiment Setup

The right image of Figure 3.2 depicts an overview of the Assistive Kitchen. The world origin is located in the lower left corner of the room. Objects that are important during the experiments are the table which is located at pose $\langle 3.1\mathrm{m}, 1.7\mathrm{m}, 0.0\mathrm{m}, 0°, 0°, 90° \rangle$, the robot's inital base position that is located at $\langle 1.0\mathrm{m}, 1.9\mathrm{m}, 0° \rangle$, and the cup which is located at $\langle 2.9\mathrm{m}, 1.7\mathrm{m}, 0.727\mathrm{m}, 0°, 0°, 270° \rangle$ with respect to the world frame.

The robot acquires experience by executing the following action sequence: 1) navigate from the initial base position to a specified position near the table; 2) reach for the object with the right arm; 3) close the gripper; 4) lift the object. In this action sequence, the task context is determined by the following parameters:

1. Type of the target object
2. Pose of the target object on the table
3. Base position from which the robot starts the reaching motion

The first task parameter is the object's type. We started by using a cup that had to be grasped at its handle. This experiment setup leads to an arm trajectory that is depicted in Figure 3.3. First, the arm moves down while concurrently rotating the endeffector so that the gripper is parallel to the cup handle. The arm continues to move down and inward, while keeping the gripper parallel to the handle. The goal pose is reached, when the handle can be grasped by closing the gripper. After successful grasping, the cup is lifted to see whether the grasp is stable or not.

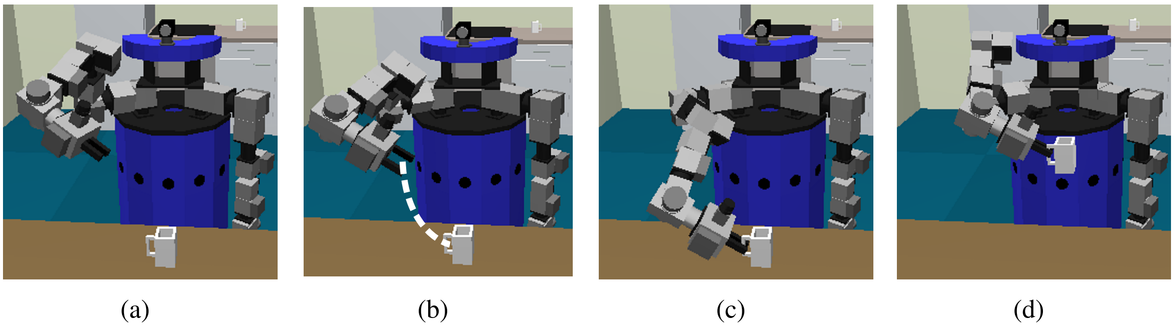

(a) (b) (c) (d)

FIGURE 3.3 a) Robot reached target base position and is ready for grasping. b) Robot opened its gripper and started to move the arm. The endeffector is already nearly parallel to the cup's handle. White dashes visualize the remaining trajectory. c) Endeffector reached the cup's handle and closed the gripper. d) Robot successfully lifted the cup.

The second task parameter is the object's pose. The cup is located in 3D space, and thus its pose (obj) can be described by a 6D vector:

$$obj = \langle x_{\text{obj}},\ y_{\text{obj}},\ z_{\text{obj}},\ \phi_{\text{obj}},\ \theta_{\text{obj}},\ \psi_{\text{obj}} \rangle$$

Please note that x_{obj} and ψ_{obj} are the only cup parameters of concern because of the following reasons. The cup is positioned on a table that is a planar surface and restricts the roll (ϕ_{obj}) and pitch angle (θ_{obj}) to 0°. The height of the target object (z_{obj}) is constrained by the height of the table, which is 0.727m. Furthermore, successful manipulation places are invariant with respect to the object's position along the table edge (y_{obj}). When the object is re-positioned by a certain distance towards the left or right table side, then the robot can adapt its base position by the same distance. This is shown in Figure 3.4, where the right image shows a cup that is shifted 0.4m to the right. When the robot adapts its base position 0.4m to the right, then it faces the same manipulation task as in the left image. So there is a linear relation between y_{obj} and manipulation place, which can be captured in an analytical model. Because there is no such simple relation from x_{obj} and ψ_{obj} to manipulation places we use experience-based learning to capture these parameters in a Generalized Success Model.

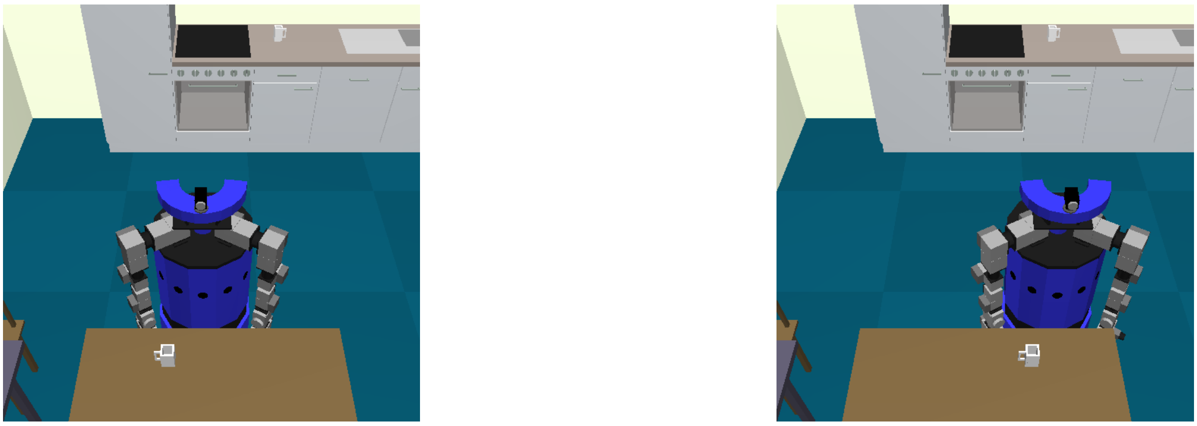


FIGURE 3.4 The cup is moved 0.4m along the table edge. When the robot changes its base position accordingly, then it faces the same manipulation task.

In order to learn x_{obj} and ψ_{obj}, the robot has to grasp the cup from 64 different cup poses, namely eight different values for the cup's distance to the long table edge (x_{obj}), and eight different angular orientations (ψ_{obj}). As depicted in the left image of Figure 3.5, x_{obj} varies from 2.80m to 3.50m with a step size of 0.1m. Higher values would not lead to successful grasping, because the robot's manipulator length is 0.84m and the table edge is located at $x_{\text{obj}} = 2.73\text{m}$. Therefore, a object that is located at $x_{\text{obj}} = 3.60m$ is out of reach. ψ_{obj} varies from 190° to 330° in a step size of 20°. The exact values for cup poses are:

$$x_{\text{obj}} \in \{2.80\text{m},\ 2.90\text{m},\ 3.00\text{m},\ 3.10\text{m},\ 3.20\text{m},\ 3.30\text{m},\ 3.40\text{m},\ 3.50\text{m}\}\ ;$$

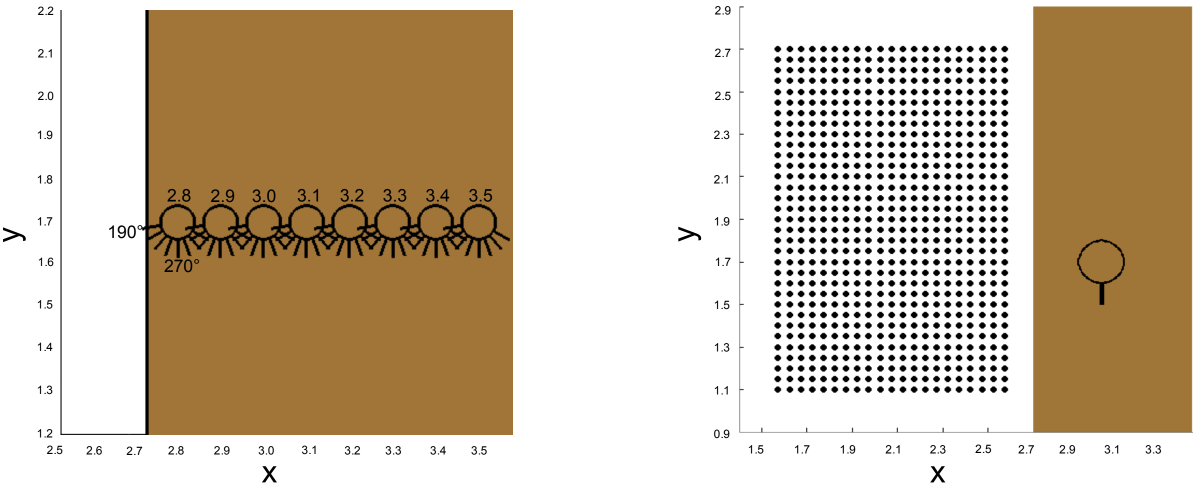


FIGURE 3.5 Left: The robot has to grasp the cup from 64 different cup poses. For clarity only cup orientations with 190° and 270° are annotated. Right: Every dot marks a base position from where the robot performs a manipulation action.

$$\psi_{\text{obj}} \in \{190^\circ,\ 210^\circ,\ 230^\circ,\ 250^\circ,\ 270^\circ,\ 290^\circ,\ 310^\circ,\ 330^\circ\}\ ;$$

$$y_{\text{obj}} = 1.7\text{m}\ ;\ \ z_{\text{obj}} = 0.727\text{m}\ ;\ \ \phi_{\text{obj}} = 0^\circ\ ;\ \ \theta_{\text{obj}} = 0^\circ\ ;$$

The third task parameter that influences grasp success is the base position from where the robot performs the grasping motion. The robot moves on a planar 2D surface, and thus its base position (rob) can be described by the vector:

$$rob = \langle x_{\text{rob}},\ y_{\text{rob}},\ \psi_{\text{rob}} \rangle$$

In order to learn successful base positions, the robot has to grasp the cup from 693 different base positions that are uniformly distributed in a rectangular area. More precisely, we use 21 different values for the robot's distance to the table and 33 values for its position along the table edge. As depicted in the right image of Figure 3.5, x_{rob} varies from 1.60m to 2.60m with a step size of 0.05m. y_{rob} varies from -0.6m to 1.0m with a step size of 0.05m. The area of robot base position is chosen loosely. Hence, further enlarging this area into any direction would not lead to successful grasping, because the robot would either be too far away from the object, or bump into the table. The exact values for robot base poses are:

$$x_{\text{rob}} \in \{1.60\text{m},\ 1.65\text{m},\ 1.70\text{m},\ 1.75\text{m},\ ..,\ 2.55\text{m},\ 2.60\text{m}\}\ ;$$

$$y_{\text{rob}} \in \{1.10\text{m},\ 1.15\text{m},\ 1.20\text{m},\ 1.25\text{m},\ ..,\ 2.65\text{m},\ 2.70\text{m}\}\ ;$$

$$\psi_{\text{rob}} \in \{0^\circ\}\ ;$$

$\psi_{\mathrm{rob}} = 0°$ because we only consider robot orientations where the robot faces the table perpendicularly.

3.2.2 Labeling Training Data

Overall, the robot has to grasp the cup at 64 different cup poses from 693 different base positions, resulting in 44.352 experiments. After each experiment, the robot logs if the manipulation action was successful, or if it failed. Figure 3.6 depicts three experiment runs. While the cup pose remains the same, the base position from where the robot starts the grasping action changes. It can be seen that grasp success is determined by the base position from where the robot starts the grasping action.

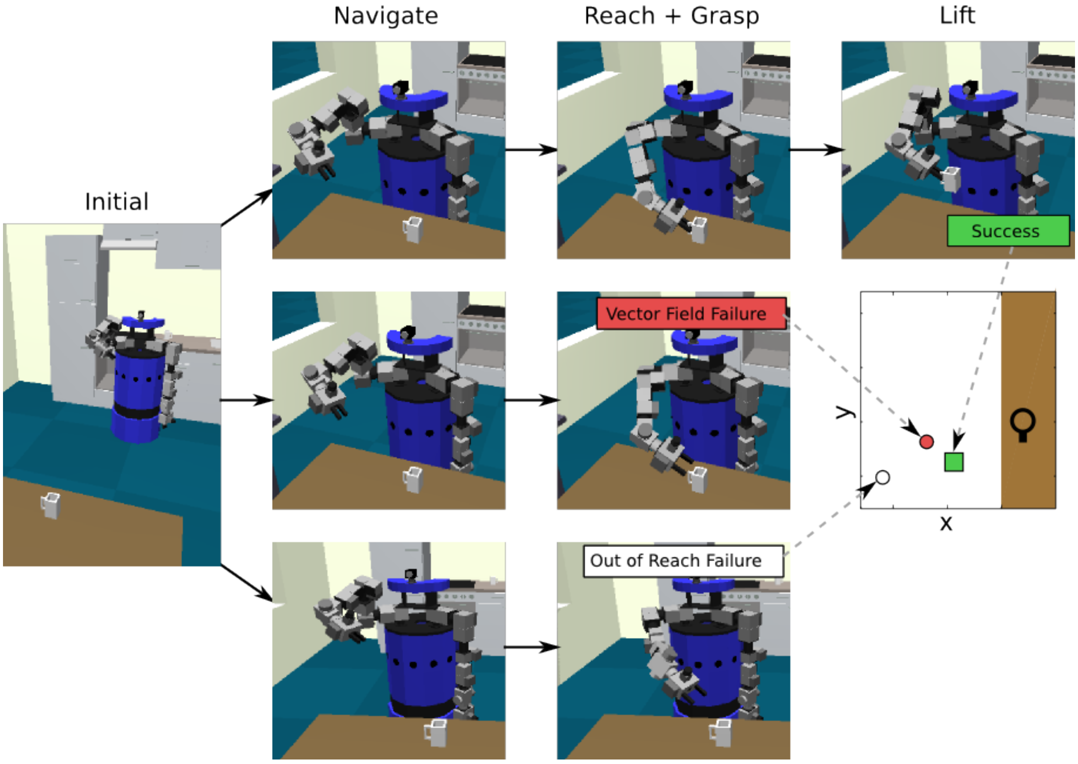


FIGURE 3.6 Three experiment runs with different samples for the robot's base position. The cup is always located at pose $\langle 2.9\mathrm{m},\ 1.7\mathrm{m},\ 0.727\mathrm{m},\ 0°,\ 0°,\ 270°\rangle$. The action sequence in the top row succeeds from base position $\langle 2.3\mathrm{m},\ 1.3\mathrm{m},\ 0°\rangle$ (green square in center right image). It fails in the center row from base position $\langle 2.1\mathrm{m},\ 1.55\mathrm{m},\ 0°\rangle$ (red circle), because the vector field controller for reaching gets stuck in a local minimum. The action sequence in the bottom row from base position $\langle 1.60\mathrm{m},\ 1.1\mathrm{m},\ 0°\rangle$ (white circle) fails, because the cup is out of reach.

The result of an experiment of whether the robot was successful at grasping the object or not is visualized by a green square, a red circle, or a white circle.

Green squares: Manipulation action successful. These are base positions from which the robot was able to successfully grasp the cup. The task execution is considered successful when the cup is more than 10cm above the table after the lift action is completed, which can only be the case if the robot is holding it.

White circles: Object theoretically unreachable. From many base positions the cup cannot be grasped. Considering that one experiment takes approximately 50 seconds to execute, performing all 44.352 experiments would last 26 days. Therefore we use analytical models to filter out experiments where we know that the robot will fail to grasp the cup before executing the experiment. This significantly speeds up the data acquisition process.

An obvious theoretical bound we implemented was that the robot's distance to the table must be at least as big as the robot's radius. Otherwise the robot would bump into the table. As the robot's radius is 0.25m, we can immediately label experiments from base positions $x_{rob} \in \{2.50\mathrm{m}, 2.55\mathrm{m}, 2.60\mathrm{m}\}$ as failures, without executing them.

From many base positions the target object cannot be grasped simply because the arm is not long enough. More formally, for these base positions, the kinematics of the arm is such that no inverse kinematics solution exists for having the end-effector at the position required to grasp the target object. The analytic model we use here is a capability map that is a compiled representation of the robot's kinematic workspace (Zacharias et al., 2007). Capability maps are usually used to answer the question: given the position of my base, which end-effector poses are reachable? Within the ARPLACE framework we use the capability map to answer the inverse question: given the position of the target object, and therefore the desired pose of my end-effector, from which base positions can I reach this end-effector position? In Figure 3.7, the answer to this question is visualized for a specific object pose. The depicted area is a theoretical kinematic upper bound on the base positions from which the robot can reach the object.

Before executing an experiment, we use the capability map to determine if the target object is theoretically reachable from the current base position. If this is not the case, we do not execute the navigate-reach-grasp action sequence, but directly label the corresponding experiment as failure (white circle).

Overall, we were able to filter out 6.336 experiments where the robot would have bumped into the table, and 30.780 experiments where the robot arm kinematics would not have led to successful grasping. As a result, we had to execute only 7.236 out of 44.352 experiments and were able to reduce the overall time for gathering training data from 26 days to 4 days.

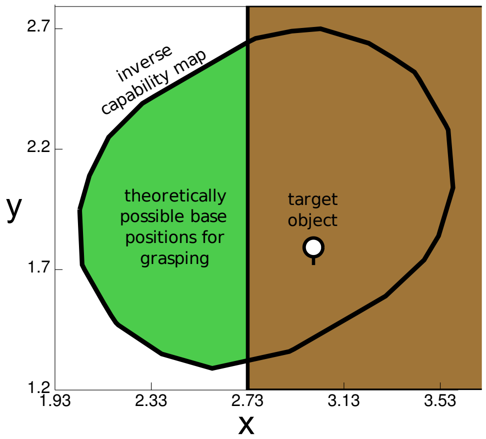


FIGURE 3.7 Inverse capability map for a specific object pose. Please note that the target object is not located in the middle of the inverse capability map. The inverse capability map is shifted to the top because it represents kinematic reachability for base positions. Base positions, however, are specified with respect to the robot's center, while the arms are mounted with an offset to the robot's center.

Red circles: Object practically unreachable. Please note that the capability map only considers the theoretical reachability from a base position, given the kinematics of the robot's arm. It does not take self-collisions into account, or the constraints imposed by our vector-field controller for reaching, or the specific hardware of our gripper, and the way the gripper interacts with the target object. Red circles represent experiments where the manipulation action is kinematically viable according to the capability map but nevertheless leads to a failure. Reasons for failurs can be that the vector field controller gets caught in a local minimum, the robot hits the cup before grasping it, the robot closes gripper without the cup handle being in it, the cup slips from the gripper after successful grasping, or the robot bumps into the table due to imprecision in the navigation routine.

3.2.3 Gathering Training Data

Figure 3.8 shows an outtake of all acquired data. The visualization illustrates that the data has some intuitive properties. First, as the cup moves further away from the table edge, more and more robot base positions are out of reach (white markers). This can be seen when examining one of the image rows from left to right. As x_{obj} increases from 2.8m to 3.1m, more white markers appear for robot positions that are far away from the table. Second, as the cup handle rotates away from the robot, the base positions from where the robot is able to successfully manipulate the cup shifts towards the table. This can be seen, when examining one of the image columns. As ψ_{obj} increases from 210° to 330°, the area of green markers shifts from the left plot side, which indicates base positions that are far away from the table, to the right side of the plot, which indicates base positions that are closer to the table. The reason is that when the cup handle points away from the robot, then the robot has to approach the handle from behind the cup's body, requiring the robot to position itself closer to the table. Moreover, a slight shift to the bottom of the plot can be observed, which makes the robot position its base more to the right of the object. This is intuitive, because the robot needs more space to the right to be able to grasp around the cup in order to approach it from behind. Third, as the cup moves away from the table edge or as the handle rotates away from the robot, the number of base positions for successful manipulation shrinks and finally reaches zero for the cup poses $\langle 3.0\text{m}, 1.7\text{m}, 0.727\text{m}, 0°, 0°, 330° \rangle$ and $\langle 3.1\text{m}, 1.7\text{m}, 0.727\text{m}, 0°, 0°, 330° \rangle$.

Please note that some of the failures surrounded by successes correspond to manipulator singularities while others are due to noise. We observed that the navigation controller is a frequent source of noise, as it is not always able to position the robot parallel to the table edge. We observed that the rotational error could be as high as 5°, which leads to an endeffector offset of up to 7cm compared to a properly aligned robot.

3.3 Computing Classification Boundaries

In this section we will use Support Vector Machines (SVMs) to generalize over the acquired training data in order to compute classification boundaries. One classification boundary is learned for each of the target object's poses. The classification boundary for a certain object pose is a polygonal region that captures robot base positions from which grasping the target object at the corresponding pose will succeed. Figure 3.9 depicts a classification boundary for cup pose $\langle 2.9\text{m}, 1.7\text{m}, 0.727\text{m}, 0°, 0°, 290° \rangle$, which corresponds to the plot in the second column of the third row in Figure 3.8.

3.3.1 Relative Feature Space

Before applying SVMs, we convert the gathered training data from global world coordinates into a relative coordinate frame. By doing so, a Generalized Success Model that is learned for a certain table position is also valid for tables with different positions and orientations. As stated previously, the base position from where an object can be grasped successfully is only determined by the target object's type, the target object's pose on the table, and the relative base positions of the robot to the target object, but not by the absolute position of the table or the robot. Figure 3.10 depicts two similar manipulation tasks. While the table's

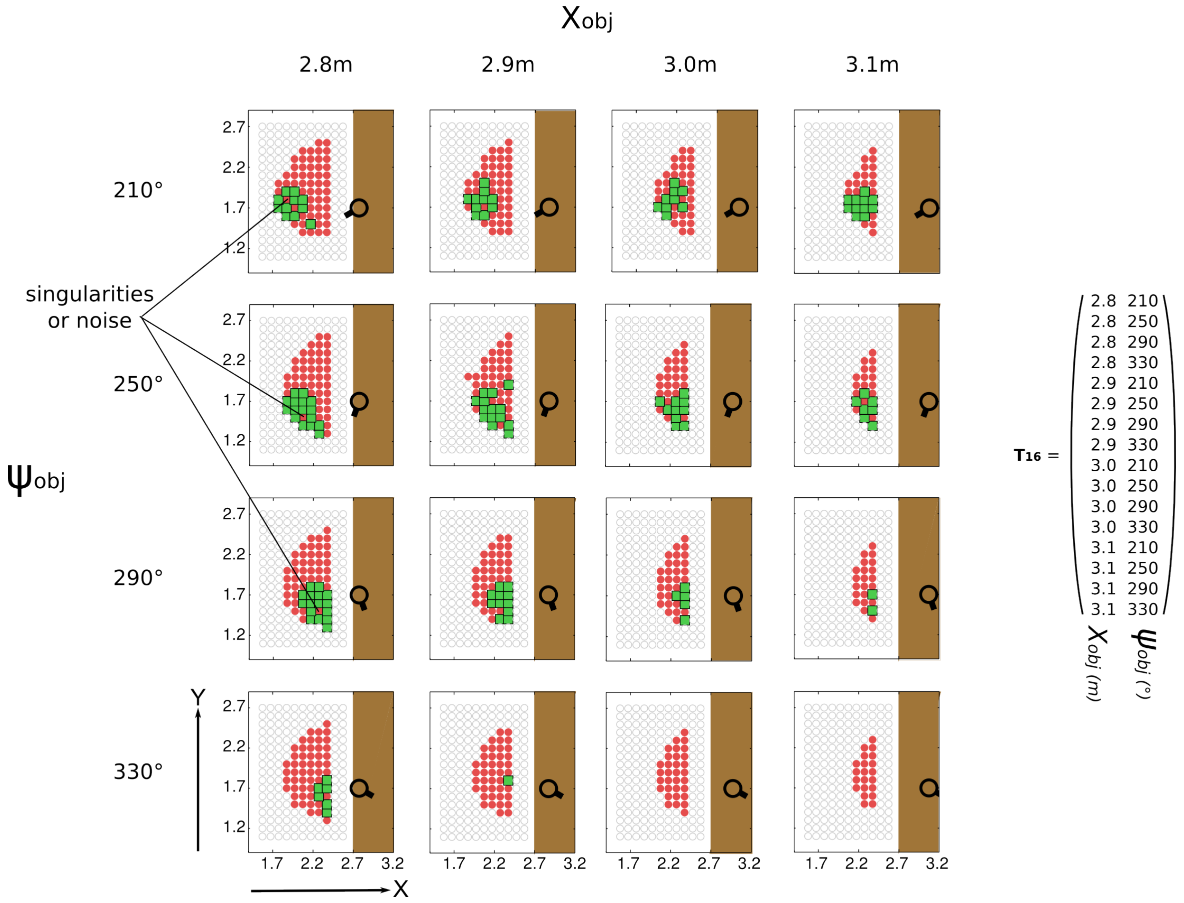


FIGURE 3.8 Every subplot shows the training data that corresponds to the cup pose that is visualized with the black cup. In every image row, cup orientation ψ_{obj} remains static while cup pose x_{obj} varies from 2.8m to 3.1m. In every image column, x_{obj} remains static while ψ_{obj} varies from 210° to 330°. Markers correspond to the center of the robot base in an experiment. Green squares and red circles represent successful and failed grasps. White circles were not executed, because a successful grasp is theoretically impossible. For clarity, the results are depicted for only 16 of the 64 object poses and only 187 of the 693 base positions. T_{16} indicates all combinations of x_{obj} and ψ_{obj} that are plotted.

position relative to the world frame changed, the pose of the cup with respect to the table frame remained the same. Therefore, when omitting external effects, then a relative position of the robot with respect to the cup that enabled the robot to successfully grasp the cup from table position 1 will also enable the robot to successfully grasp the cup from table position 2 when the cup's pose relative to the table frame remains static.

The origin of the relative feature space is defined by the table's edges and the pose of the cup on the table. First, we compute the normal $\overrightarrow{n_{TE}}$ from the object to the table edge $\overrightarrow{TE}$ that is closest to the robot. The orientation of $\overrightarrow{TE}$ is chosen so that the table is on the left side of the vector. The origin of the relative feature space is located at the intersection of $\overrightarrow{n_{TE}}$ and $\overrightarrow{TE}$,

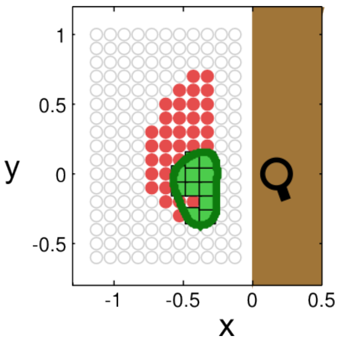


FIGURE 3.9 Classification boundary for cup pose $\langle 2.9\text{m}, 1.7\text{m}, 0.727\text{m}, 0°, 0°, 290° \rangle$.

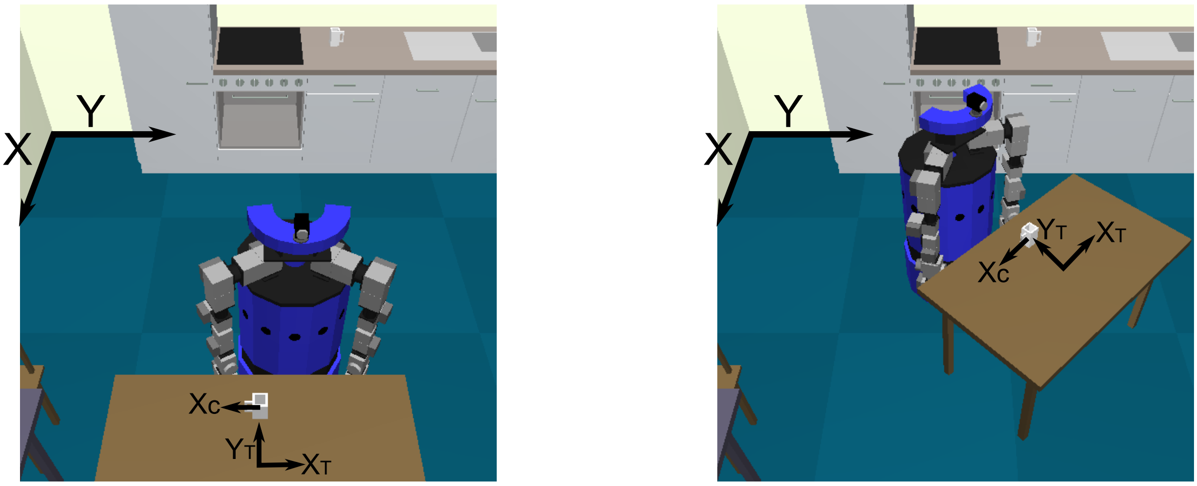


FIGURE 3.10 Two similar manipulation tasks. While the global position of the table changed from $\langle 3.1\text{m}, 1.7\text{m}, 0.727\text{m}, 0°, 0°, 90° \rangle$ to $\langle 2.1\text{m}, 2.2\text{m}, 0.727\text{m}, 0°, 0°, 135° \rangle$, the relative pose of the cup with respect to the table remained the same $\langle 0.0\text{m}, 0.2\text{m}, 0.0\text{m}, 0°, 0°, 180° \rangle$. The depicted frames are the world frame, the table frame T, and the cup frame C. All frames are right-handed with the z-axis pointing upward. For clarity, Y_c is not depicted but can be deduced from the other two axis.

as depicted in the left image of Figure 3.11. The z-axis of the relative feature space Δz points upward. Δx points towards the object, so it is identical to $\overrightarrow{n_{TE}}$ but rotated by 180° around Δz. Δy is defined by the other two axis, and in the left image of Figure 3.11 it is identical to $\overrightarrow{TE}$ but rotated by 180° around Δz.

Now we are able to compute the poses of the robot and the object with respect to the relative feature space, as depicted in the right image of Figure 3.11. The robot's relative position is defined through Δx_{rob} (distance from the robot's base to the table edge) and Δy_{rob} (distance of the robot's base on the table edge). In the following, we call Δx_{rob} and Δy_{rob} *controllable parameters*, because the robot can change them as he likes by moving around.

The object's relative pose is defined through Δx_{obj} and $\Delta \psi_{\text{obj}}$. Please note that Δy_{obj} is always 0.0m, because we chose the origin of the relative feature space to assure that. Besides Figure 3.4, this is another explanation why we are able to avoid additional experiments by varying the object's y-pose. In the following, we will call Δx_{obj} and $\Delta \psi_{\text{obj}}$ *task-relevant parameters*. The robot can estimate these parameters with its vision system, but can influence them only by performing a manipulation action. More specific, the robot can choose the pose from where to perform the grasp, but it cannot choose the pose of the target object.

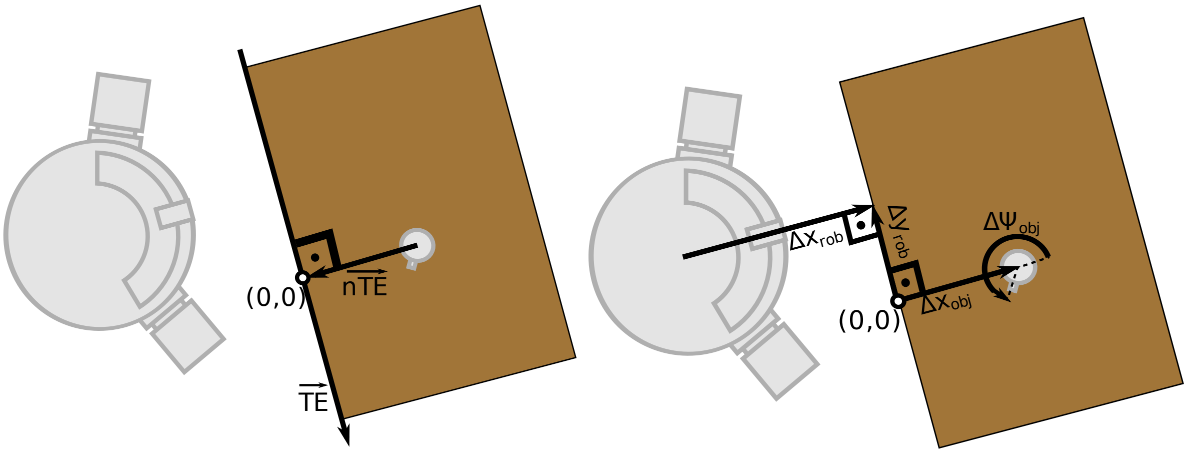


FIGURE 3.11 Left: Finding the origin of the relative feature space. It is labeled with $(0, 0)$. The axis of the relative feature space are not shown for clarity, but Δz points upward, Δx is identical to $\overrightarrow{n_{te}}$ but rotated by 180° around Δz, and Δy is identical to $\overrightarrow{TE}$ but rotated by 180° around Δz. Right: Relative feature space with controllable parameters Δx_{rob} and Δy_{rob}, and observable parameters Δx_{obj} and $\Delta \psi_{\text{obj}}$.

The poses of the cup and the robot that were acquired in the data gathering process were relative to the world frame. When we transform these poses into poses that are relative to the relative feature space, we get:

$$\Delta x_{\text{rob}} \in \{\text{-}1.13\text{m}, \text{-}1.08\text{m}, \text{-}1.03\text{m}, \text{-}0.98\text{m}, .., \text{-}0.18\text{m}, \text{-}0.13\text{m}\};$$

$$\Delta y_{\text{rob}} \in \{\text{-}0.60\text{m}, \text{-}0.55\text{m}, \text{-}0.50\text{m}, \text{-}0.45\text{m}, .., 0.95\text{m}, 1.00\text{m}\};$$

$$\Delta \psi_{\text{rob}} = 0^\circ \ ;$$

$$\Delta x_{\text{obj}} \in \{0.07\text{m}, 0.17\text{m}, 0.27\text{m}, 0.37\text{m}, 0.47\text{m}, 0.57\text{m}, 0.67\text{m}, 0.77\text{m}\} \ ;$$

$$\Delta \psi_{\text{obj}} \in \{190^\circ, 210^\circ, 230^\circ, 250^\circ, 270^\circ, 290^\circ, 310^\circ, 330^\circ\} \ ;$$

$$\Delta y_{\text{obj}} = 0.0\text{m} \ ; \ \ \Delta z_{\text{obj}} = 0.727\text{m} \ ; \ \ \Delta \phi_{\text{obj}} = 0^\circ \ ; \ \ \Delta \theta_{\text{obj}} = 0^\circ \ ;$$

3.3.2 Computing Classification Boundaries

A classification boundary is a model that maps an input with two parameters to a boolean value. In our case the input vector is the robot's relative base position $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle$ when starting a manipulation action. The resulting boolean value is the classification boundary's prediction whether the manipulation action will succeed or fail from there. Therefore, a classification boundary implements a mapping

$$f : \mathbb{R} \times \mathbb{R} \rightarrow \{0, 1\}$$

where $f(\Delta x_{\text{rob}}, \Delta y_{\text{rob}}) = 1$ if the manipulation action succeeds when performed from base position $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle$, and $f(\Delta x_{\text{rob}}, \Delta y_{\text{rob}}) = 0$ if it fails. Figure 3.12 depicts a classification boundary for a particular cup pose.

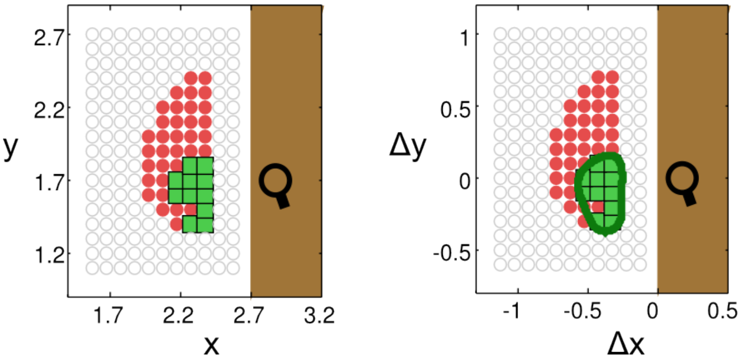


FIGURE 3.12 Left: Gathered training data with respect to the world frame for cup pose $\langle 2.9\text{m},\ 1.7\text{m},\ 0.727\text{m},\ 0°,\ 0°,\ 290° \rangle$. Right: Training data with respect to the relative feature space. The dark green hull is the classification boundary that was learned with support vector machines.

It is a polygonal region, where $f(\Delta x, \Delta y) = 1$ when $(\Delta x, \Delta y)$ is inside the classification boundary, and $f(\Delta x, \Delta y) = 0$ otherwise. Applied to our scenario, a classification boundary is a polygonal region that represents robot base positions for successfully manipulating a target object that is located at a certain pose. Therefore, the classification boundary has to find regions that contain as many green markers as possible, and exclude as many red and white markers as possible. We learn such classification boundaries by using *support vector machines*. In principle, any binary classification algorithm can be used to learn classification boundaries, but support vector machines have several properties that make them especially suitable for the problem at hand.

First, support vector machines compute the maximally separating hyperplane or set of hyperplanes. This means that the hyperplanes are chosen in order to maximize the distance to the nearest training datapoints of any class – in our case these classes are 'successes' and 'failures'. Choosing maximally separating hyperplanes has the advantage that the learned model generalizes well. In contrast to support vector machines, other classification algorithms like neural nets and decision trees try to find any separating hyperplane.

Second, we prefer to obtain smooth classification boundaries, because the point distribution model that is used to generalize over classification boundaries works better when classification boundaries are smooth. Support vector machines have the property of creating smooth boundaries. The support vector machine implementation we use even has a parameter that allows to influence boundary smoothness.

Third, support vector machines are easy to use and scale well. Support vector machines use kernel methods that map the input space into a higher dimensional state space in which the problem is linearly separable. This process is called kernel trick and can be computed efficiently. As a result, support vector machines find promising state spaces by themselves. In other classification algorithms, the algorithm designer has to analyse the domain very carefully in order to define the state space (input parameters) in which the classification takes place. Manually designing the state space is time consuming, error prone, does not scale well, and might lead to state spaces that are inferior.

The SVM implementation we use is called "Shogun" and was implemented by Sonnenburg et al. (2006). It is freely available as open source code. Shogun allows to customize support vactor machine by specifying a kernel, and supports the creation of combined kernels which can be constructed by a weighted linear combination of several sub-kernels. For the rest of this thesis, we will use a Gaussian kernel. A cost parameter C controls the trade off between allowing training errors and forcing rigid margins. This means that it is a parameter that can be tweaked to make the classification more tight but prone to overfitting (higher values for C), or more loose but at the risk of incorrectly labeling large amounts of data (lower values for C). Lower values for C will also result in smoother classification boundaries. For the rest of the thesis we choose C to be 40.0, which we consider to be a good tradeoff.

As input for support vector machines, we label green markers as '1' (success), and red and white markers as '0' (failure). Because we want to find classification *boundaries*, we have to ensure that regions of grasp success are surrounded by failed experiments towards all directions. That is why we chose x_{obj} and y_{obj} generously, ensuring that theoretically unreachable base positions will occur towards every direction for every target object pose. For example, we included $x_{\text{obj}} \in \{2.50\text{m}, 2.55\text{m}, 2.60\text{m}\}$, although we knew that the robot will bump into the table before performing the experiment (compare Figure 3.12). As successful grasps are rarer, we weight them twice as much as failed grasp attempts.

Because classification boundaries capture regions of successful grasping only for a certain object pose, one classification boundary has to be learned for every target object pose. In our case we have to learn 64 classification boundaries. Figure 3.13 depicts 16 out of 64 classification boundaries that were learned for the training data that was depicted in Figure 3.8.

The classification boundaries visualize some intuitive properties. First, as the cup moves further away from the table edge, the corresponding classification boundary approaches the table edge. This can be seen when watching one of the image rows from left to right. While the shape of classification boundaries remains relatively constant for cup poses near the table edge, the shape changes when the cup's distance to the table edge exceeds a certain threshold. The reason is that the robot can compensate movements of the cup towards the table center by moving closer to the table by the same distance. However, if the robot's body reaches the table edge, then this compensation is not possible anymore, because the robot would bump into the table. That is why the classification boundary gets clipped at its front, when the cup's distance to the table edge exceeds a certain threshold. Second, as the cup handle rotates away from the robot, the classification boundaries shift. When $\Delta\psi_{\mathrm{obj}}$ increases from 210° to 330°, then classification boundaries shift closer to the table. This can be seen when watching one of the image columns from top to bottom. Moreover, a slight shift to the right of the table can be observed, which is due to the robot requiring more space in order to grasp around the cup.

3.3.3 Evaluation of learned Classification Boundaries

In the following, we evaluate if the support vector machine was able to learn classification boundaries, that precisely represent the structure of successful manipulation places. The evaluation was performed as follows. First, we partitioned the training data into two sets S_1 and S_2. S_1 contained $\frac{1}{3}$ and S_2 contained $\frac{2}{3}$ of the data. Then a support vector machine with a Gaussian kernel and the parameter C set to 40.0 was used to learn classification boundaries. The resulting classification boundaries were used to classify the data from set S_2. The result was that 95% of the data was labeled correctly, which indicates that overfitting is well under control.

In a second experiment, we evaluated if no underfitting occured either. Therefore, we learned classification boundaries on the whole training data. When using the classification boundaries to classify the training data, 97% of the data was labeled correctly. This indicates that no underfitting occured and suggests that the parameter C was chosen in a meaningful way.

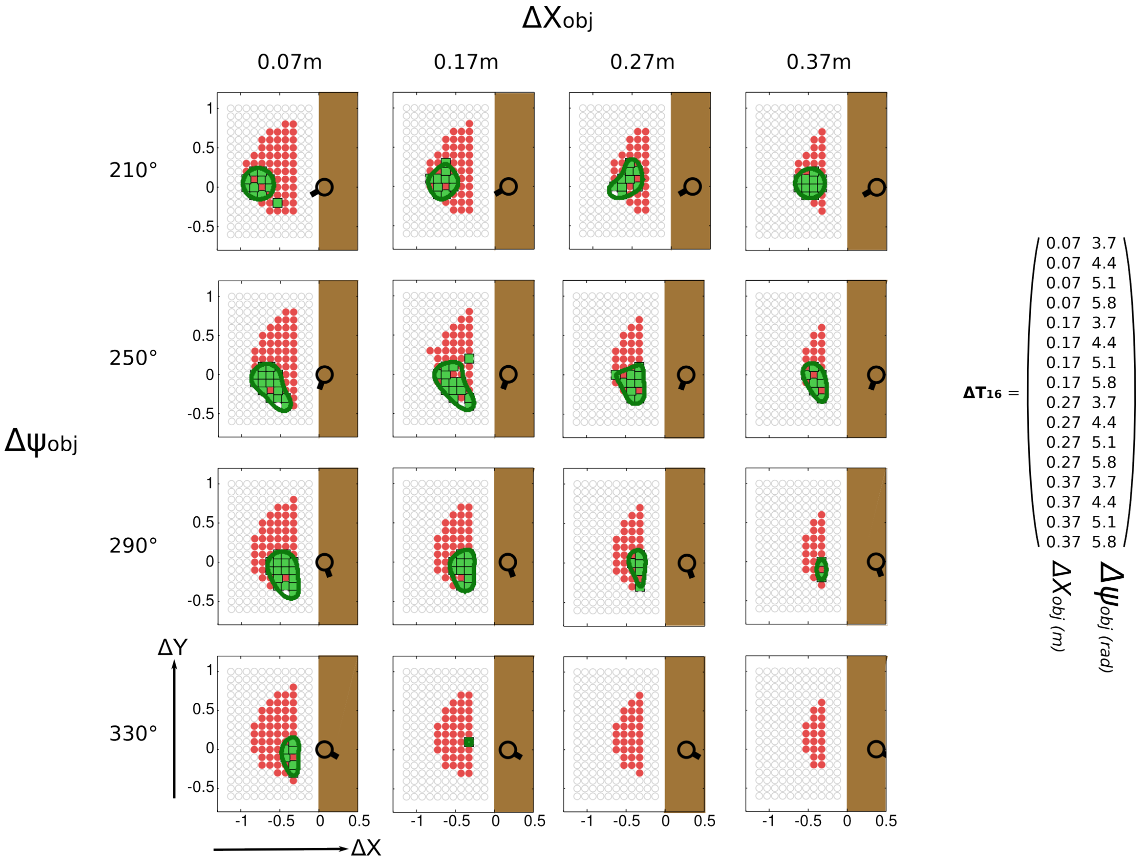


FIGURE 3.13 Classification boundaries for the training data that was depicted in Figure 3.8. Coordinates are specified with respect to the relative feature space.

3.4 Generalization over Classification Boundaries

Classification boundaries capture polygonal regions for successfully manipulating an object at a particular pose. A robot could use classification boundaries to compute promising base positions. However, classification boundaries cannot interpolate between object poses. When the robot estimated an object's pose, it would have to find the corresponding classification boundary even if no classification boundary was learned for the observed object pose. A naive approach would be to find the closest object pose for which a classification boundary was learned. A loss of accuracy however, would be inevitable. Moreover, the impossibility of interpolating object poses would prevent to take state estimation uncertainties into account, as will be explained in chapter 4.

In this chapter we will generalize over classification boundaries and compile them into a single point distribution model (PDM). The PDM is more compact than classification boundaries, supports the interpolation between object poses, and can be computed quickly.

3.4.1 Aligning Classification Boundaries

A PDM requires n landmarks as input that are distributed over a contour. We distribute 20 landmarks equidistantly over each classification boundary, and determine the correspondence between landmarks on different boundaries by minimizing the sum of the squared distances between corresponding landmarks, while maintaining order between the landmarks on the boundary. We essentially perform a Procrustes analysis, and at the same time determine the optimal position of the landmarks.

Figure 3.14 depicts the process of aligning classification boundaries. In the top left image, two classification boundaries from Figure 3.13 are drawn. More specific, the classification boundaries are the ones that correspond to object pose $\langle \Delta x_{\mathrm{obj}}, \Delta \psi_{\mathrm{obj}} \rangle = \langle 0.07\mathrm{m}, 210°\rangle$ (black color) and $\langle \Delta x_{\mathrm{obj}}, \Delta \psi_{\mathrm{obj}} \rangle = \langle 0.07\mathrm{m}, 250°\rangle$ (dark green color). The next step is to distribute 20 landmarks equidistantly on both boundaries, which is depicted in the top right image. Every fifth landmark is numbered. There is no particular rule, where the numbering has to be started. Landmark 1 of the black classification boundary is at its top, while landmark 1 of the dark green classification boundary is at the lower left.

The bottom left image shows how the classification boundaries are aligned by aligning their landmarks. The rule is to find correspondences between landmarks on different classification boundaries is twofold.

- minimize the sum of squared distances between corresponding landmarks on different boundaries
- maintain order between landmarks and try to maximize the distance between landmarks on the same boundary

Regions with many landmarks indicates an area where minimizing the sum of squared distances between corresponding landmarks is the dominant factor. Regions with few landmarks indicate an area where maximizing the distance between landmarks on the same boundary is more important. The bottom right image depicts four aligned classification boundaries. The aligned classification boundaries correspond to the four classification boundaries of Figure 3.13 where $\Delta x_{\mathrm{obj}} = 0.07\mathrm{m}$ (first image column). Landmark 2 on classification boundary 1 (black color) is colored red. The position of this landmark is $\langle \Delta x_{\mathrm{c1\_l2}}, \Delta y_{\mathrm{c1\_l2}} \rangle = \langle -0.67\mathrm{m}, 0.20\mathrm{m}\rangle$, and also colored red.

3.4.2 Point Distribution Model

Given the aligned landmarks on the classification boundaries we are able to compute a point distribution model. Although PDMs are most well-known for their use in computer vision

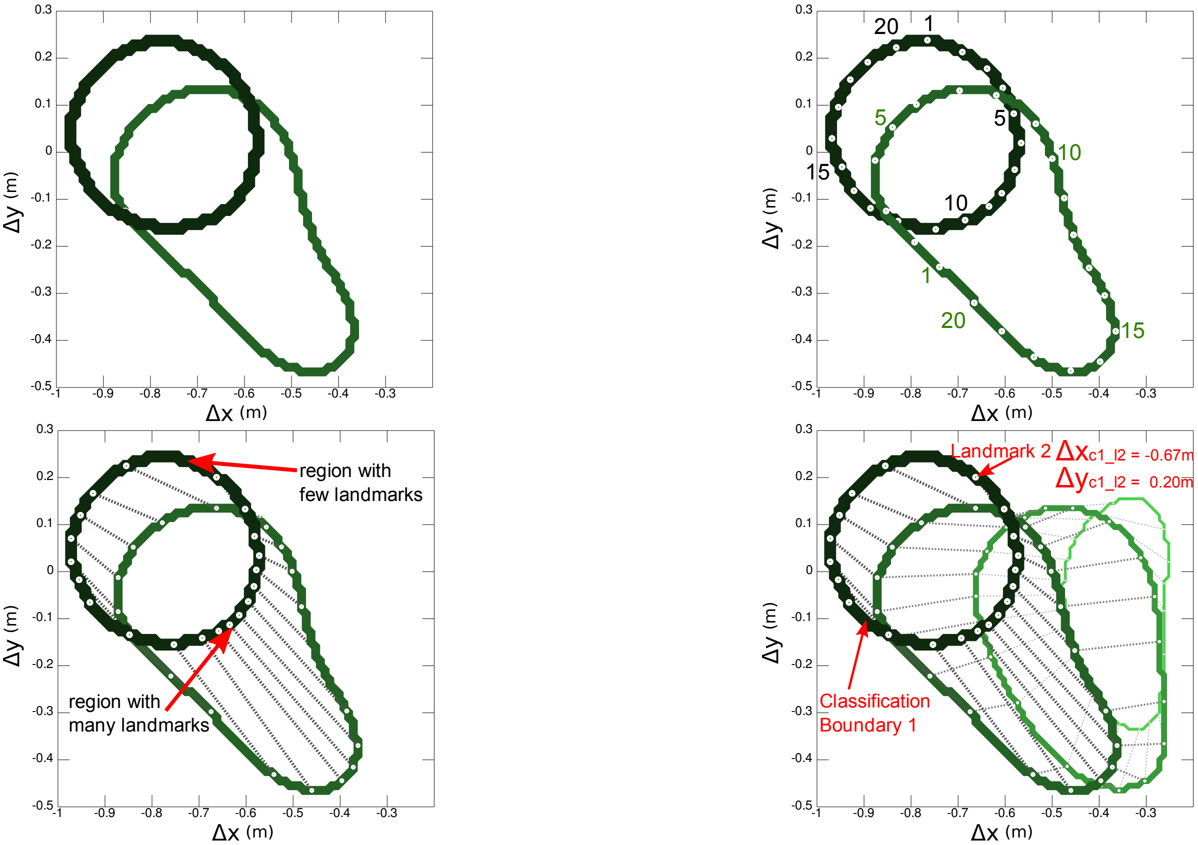


FIGURE 3.14 Process of aligning classification boundaries by aligning landmarks that are distributed on the classification boundary. A detailed description is presented in the text. Lines indicate that the corresponding landmarks are aligned.

systems that analyse medical images (Cootes et al., 1995a) and faces (Wimmer et al., 2008), we use the notation by Roduit et al. (2007) that focusses on robotic applications. First, the 64 classification boundaries are merged into one 40x64 matrix **H**, where the n.th column is the concatenation of the Δx and Δy coordinates of the 20 landmarks along the classification boundary. Each column thus represents one classification boundary.

$$H = \begin{pmatrix} \Delta x_{c1\_l1} & \Delta x_{c2\_l1} & \Delta x_{c3\_l1} & \ldots & \Delta x_{c63\_l1} & \Delta x_{c64\_l1} \\ \Delta y_{c1\_l1} & \Delta y_{c2\_l1} & \Delta y_{c3\_l1} & \ldots & \Delta y_{c63\_l1} & \Delta y_{c64\_l1} \\ \Delta x_{c1\_l2} & \Delta x_{c2\_l2} & \Delta x_{c3\_l2} & \ldots & \Delta x_{c63\_l2} & \Delta x_{c64\_l2} \\ \Delta y_{c1\_l2} & \Delta y_{c2\_l2} & \Delta y_{c3\_l2} & \ldots & \Delta y_{c63\_l2} & \Delta y_{c64\_l2} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \Delta x_{c1\_l20} & \Delta x_{c2\_l20} & \Delta x_{c3\_l20} & \ldots & \Delta x_{c63\_l20} & \Delta x_{c64\_l20} \\ \Delta y_{c1\_l20} & \Delta y_{c2\_l20} & \Delta y_{c3\_l20} & \ldots & \Delta y_{c63\_l20} & \Delta y_{c64\_l20} \end{pmatrix}$$

The next step is to compute $\mathbf{P}$, which is the matrix of eigenvectors of the covariance matrix of $\mathbf{H}$. $\mathbf{P}$ represents the principal modes of variation. Given $\overline{\mathbf{H}}$ which is the mean of all classification boundaries, and $\mathbf{P}$, each classification boundary $\mathbf{h}_{1..64}$ can be decomposed into the mean boundary and a linear combination of the columns of $\mathbf{P}$ as follows $\mathbf{h}_k = \overline{\mathbf{H}} + \mathbf{P} \cdot \mathbf{b}_k$. Here, $\mathbf{b}_k$ is the *deformation mode* of the k^{th} classification boundary. This is the point distribution model. To get an intuition for what the PDM represents, the first three deformation modes are depicted in Figure 3.15. In the left image, the values of the first deformation mode is varied between the maximal and minimal value, whilst the other deformation modes are set to 0. In the center image, the second deformation mode is varied, and the right image depicts variation of the third deformation mode.

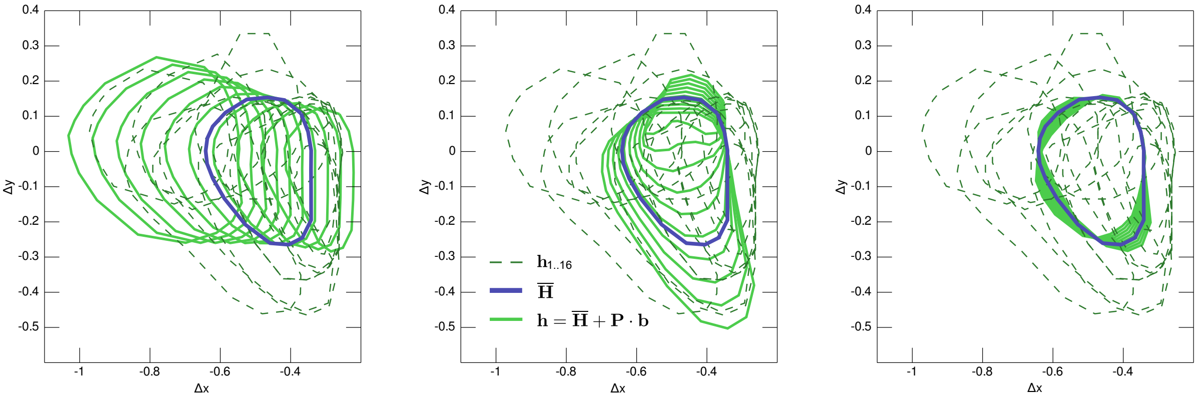


FIGURE 3.15 The first three deformation modes of the PDM. Variation of the first (left plot), second (center plot), and third deformation mode (right plot). Dashed green boundaries: 16 classification boundaries. They are the same in each plot. Blue boundary: Mean classification boundary. Solid green boundaries: Deformation modes of the learned PDM.

The eigenvalues of the covariance matrix of $\mathbf{H}$ indicate that the first two deformation modes contain 96% of the deformation energy. As a result, the first two deformation modes are already a model that precisely captures the shapes of all classification boundaries. This becomes obvious when watching the plots in Figure 3.15. While the left and center plot add significant deformation information, the right plot does not add a lot of information any more. It is mostly located around the mean deformation mode. For reasons of compactness and in order to achieve better generalization, we use only the first two deformation modes without losing much accuracy.

The advantage of the PDM is not only that it substantially reduces the high dimensionality of the initial 40D classification boundaries. The PDM also allows to interpolate between initial boundaries in a principled way using only two deformation parameters. The PDM is therefore a compact and general, yet accurate model for the classification boundaries that we call Generalized Success Model.

3.4.3 Relation to Task-Relevant Parameters

The final step of model learning is to acquire a mapping from task-relevant parameters $\mathbf{T}$ (object poses) that are varied during data gathering to the specific deformation of each boundary: $\mathbf{B} = f(\mathbf{T})$. Here, $\mathbf{T}$ contains the 64 relative coordinate combinations of cup poses $\langle \Delta x_{\text{obj}}, \Delta\psi_{\text{obj}} \rangle$. 16 of them are depicted in Figure 3.13 as ΔT_{16}. For this mapping, we apply a second order polynomial regression model, because it yields high coefficients of determination of $R^2 = 0.99$ and $R^2 = 0.96$ for the first and second deformation modes respectively. The coefficients of the polynomial model are stored in two 3x3 upper triangular matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, such that $\mathbf{B} \approx [\ diag([\mathbf{T}\ \mathbf{1}] \cdot \mathbf{W}_1 \cdot [\mathbf{T}\ \mathbf{1}]^T)\ diag([\mathbf{T}\ \mathbf{1}] \cdot \mathbf{W}_2 \cdot [\mathbf{T}\ \mathbf{1}]^T\ ]$. The Generalized Success Model now consists of

- 1) $\overline{\mathbf{H}}$, the mean of the classification boundaries
- 2) $\mathbf{P}$, the principal modes of variation of the classification boundaries
- 3) $\mathbf{W_{1,2}}$, the mapping from task-relevant parameters to deformation modes

Given a *novel* relative position of the cup on the table $\mathbf{t}^{new} = \langle \Delta x^{new}_{\text{obj}}, \Delta\psi^{new}_{\text{obj}} \rangle$, the Generalized Success Model estimates the area for successful grasping as follows. First, the appropriate deformation values from the cup pose are computed with $\mathbf{b}^{new} = [\ \mathbf{q} \cdot \mathbf{W_1} \cdot \mathbf{q}^T\ \ \mathbf{q} \cdot \mathbf{W_2} \cdot \mathbf{q}^T\ ]$, where $\mathbf{q} = [\mathbf{t}^{new}\ 1]$. Then the boundary is computed with $\mathbf{h}^{new} = \overline{\mathbf{H}} + \mathbf{P} \cdot \mathbf{b}^{new}$. If the robot's base position $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle$ is within the boundary $\mathbf{h}^{new}$, then the model predicts that the robot will be able to successfully grasp the object. Please note that this prediction can be made very quickly because it requires less than ten simple matrix operations.

To summarize the estimation of successful manipulation places: given the task-relevant parameters in a situation, the algorithm computes the deformation modes. The better deformation modes are used to reconstruct a classification boundary, which predicts from which base positions manipulation will succeed. This approach adheres to the proposed strategy of "learning task-relevant features that map to actions, instead of attempting to reconstruct a detailed model of the world with which to plan actions" (Kemp et al., 2007).

An overview of all steps that are required in order to learn a Generalized Success Model is depicted in Algorithm 1.

```
input  : $\mathbf{T}$ ;                                      (task relevant parameters (object poses))
         $\mathbf{C}$ ;                                  (controllable parameters (robot positions))
output : gsm ;                                                  (generalized success model)
forall $object_{x\psi}$ $in$ $\mathbf{T}$ do
    experience_set.clear( );
    forall $robot_{xy}$ $in$ $\mathbf{C}$ do
        success? = executescenario($robot_{xy}$, $object_{x\psi}$);
        experience_set.add( $\langle$object$_{x\psi}$, robot$_{xy}$, success?$\rangle$ );
    end
    experience_set_rel = transform(experience_set) ;    (transform to rel. feature space)
    boundary = classify(experience_set_rel) ;                                 (with SVM)
    boundary_set.add( $\langle$object$_{x\psi}$, boundary$\rangle$ );
end
$\mathbf{H}$ = alignpoints(boundary_set);
$\langle \overline{\mathbf{H}}, \mathbf{P}, \mathbf{B} \rangle$ = computePDM($\mathbf{H}$);
$\mathbf{W} = [\mathbf{1}\ \mathbf{T}]/\mathbf{B}^T$ ;            (mapping from task relevant parameters to $\mathbf{B}$)
gsm = $\langle \overline{\mathbf{H}}, \mathbf{P}, \mathbf{W} \rangle$
```

Algorithm 1: Computing a Generalized Success Model

3.4.4 Performance Analysis of Learning Generalized Success Models

The runtime for learning a Generalized Success Model for a training set with 44.352 experiments is presented in Figure 3.16.

Computational Step	**Time** (in seconds)
1. Learn Classification Boundaries	8.0
2. Align Classification Boundaries	158.7
3. Learn Point Distribution Model	36.5
4. Relation to Task-Relevant Parameters	0.03
Overall	**203.2**

FIGURE 3.16 Runtime for learning a Generalized Success Model.

With 203.2 seconds learning a Generalized Success Model is not realtime. We do not consider this to be problematic because it can be done offline and has to be done only once. It can be seen that the runtime is dominated by aligning the classification boundaries. For the complete training set an overall of 140.185.600 distances between landmarks had to be computed in order to obtain an optimal distribution of landmarks across all classification boundaries.

3.5 Human Activity Data for Biased Exploration

This section presents an approach for further reducing the number of experiments that are required in order to learn Generalized Success Models. As can be seen in Figure 3.13, only the data points that are near to a classification boundary actually contribute to the classification boundary's shape. Therefore, it is advantageous to find the transition from successful to failed examples with as little experiments as possible. The idea is to analyse the places from which human people perform manipulation actions and use this knowledge in order to bias the robot's exploration process. Of course this is only possible when we can reasonably assume that the robot's and human's places are similar. This is the case for our B21r mobile robot because its manipulator is similar to a human arm when considering joint lengths. The next two paragraphs explain work that has been implemented by other members of the chair.

Bandouch and Beetz (2009) developed a markerless tracking system at our chair that is able to observe humans in our Assistive Kitchen environment while they perform mobile manipulation tasks. Video data from four ceiling-mounted cameras is used as input. The system is capable of tracking human manipulation actions with a 51 DOF articulated human model, as depicted in the left and center image of Figure 3.17. Markerless tracking is unintrusive and unconstrained, enabling the humans to perform as natural as possible. The video data is complemented by object detections from Radio Frequency IDentification (RFID) sensors. For example, a RFID sensor under the table detects known objects and stores the time when they were placed on or removed from the table. The acquired data is published in the publicly available TUM Kitchen Data Set, which is described by Tenorth et al. (2009).

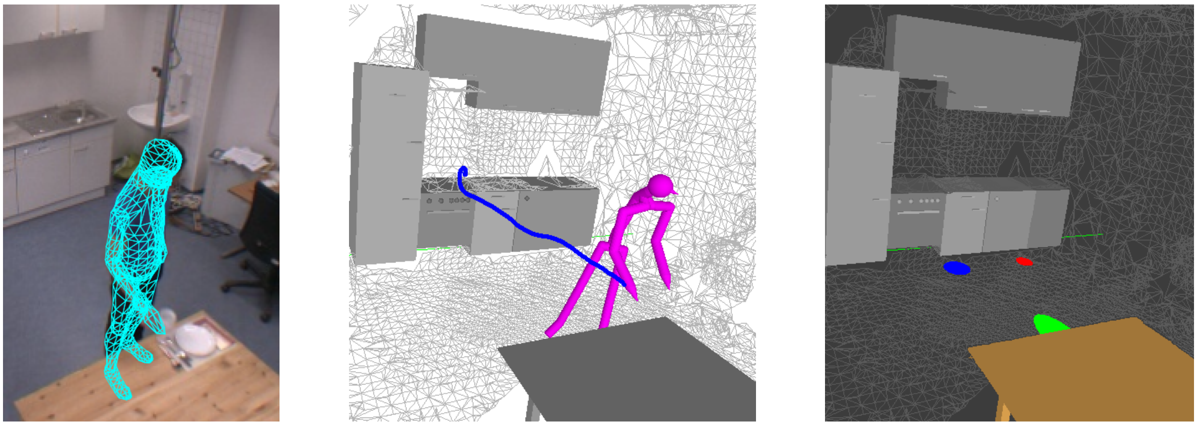


FIGURE 3.17 Left: A markerless fullbody tracking system observes a human that places a cup on the table. Center: Playback of the recorded action trajectory. Right: The circles correspond to human actions of putting an object down onto the table (green), into the sink (blue), or into the dishwasher (red).

To use the data in order to guide exploration, the system has to select relevant human poses from the continuous stream of tracked motions. First, the observed positions are loaded into a knowledge processing system and clustered with respect to their Euclidean distance. These clusters are represented as “places” in a knowledge base. Then, based on the RFID tag detections, the system learns a mapping of action properties to human “places”. Given such a model, it is possible to either obtain a place for an action, like a place for picking up a cup, or to find the most probable action when an observation is given. The right image of Figure 3.17 depicts an example of learned action places in the knowledge base, with the circle in the front marking places where humans are standing when putting objects onto the table, into the sink, or into the dishwasher. A more detailed description of the knowledge processing system and the acquisition of action models as abstract specifications of action-related concepts is given by Tenorth and Beetz (2009). Summarizing, the human activity data is a model of places from which humans perform manipulation actions.

The gray area in the left image of Figure 3.18 depicts places that humans use for manipulation. It can be seen that the area is extremely dense, measuring just about $10\mathrm{cm} \times 10\mathrm{cm}$. Although the training set was too small to provide statistically significant results, the assumption is admissible that humans have a precise, low-variance model of promising manipulation places.

The idea for learning a Generalized Success Model with fewer experiments is to use the human model as a bias for exploration. Therefore, the robot randomly samples base positions that have different distances to the area of human manipulation places. The robot starts with base positions that are in the area of human manipulation places, and continues to sample base positions that are more far away. Because the kinematics of the B21r’s manipulators is similar to that of humans regarding link lengths we expected that the robot would be able to grasp the target object when manipulation is performed from the same places that humans use for manipulation. As the distance to human places increases we expected that more and more failures would occur, until the distance is above a certain threshold and only failures would occur. When this happens, we have the boundary that separates successes from failures and support vector machines can learn the coresponding classification boundary. With this exploration strategy we avoid wasting additional time for performing experiments in areas that lead to failure, although the capability map considers them to be kinematically possible.

The left plot of Figure 3.18 visualizes the results of this exploration strategy. It can be seen that successful manipulation (green squares) indeed occurs when the robot performs manipulation actions from places that are within or near human manipulation places. As the distance increases more manipulation actions fail (red dots). The distance from manipulation places to the human model is visualized by black isobars that are drawn in increments of 5cm. The robot starts with randomly sampling base positions within the convex hull that represents human manipulation places, then within the first isobar, then within the second isobar and so on. Exploration is stopped when all randomly sampled base positions within an isobar fail, because we then know that successful experiments are completely surrounded by failures. Remember that this has been the precondition for learning classification boundaries. In the left image of Figure 3.18 it can be seen that exploration is stopped after exploring manipulation places in six isobars.

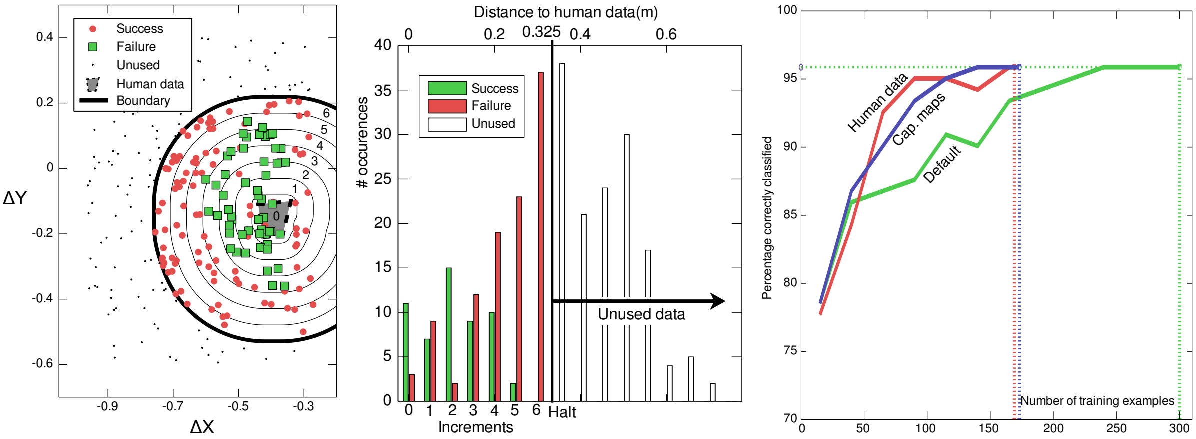


FIGURE 3.18 Left: The gray area visualizes places from where humans perform manipulation actions. Green squares, red dots and black points visualize experiment results of robot manipulation when using human data as bias. The black circles are isolines of equal distance to the convex hull around human manipulation places. Center: Histogram that depicts the number of successful (green) and failed (red) grasp attempts dependent on the distance of the robot's base to human manipulation places. Right: Quality of GSM according to exploration strategy and number of experiments.

The center plot of Figure 3.18 depicts a histogram that plots for every isobar the number of successful (green bars) and failed (red bars) manipulation places. It can be seen that there are two successful manipulation places within isobar five, but none within isobar six. Isobar 7 and onwards are not explored any more.

The right image of Figure 3.18 plots the quality of the Generalized Success Model for several exploration strategies with respect to the number of used training data. Three exploration strategies are plotted. The default strategy performs grid-based exploration by trying out all base positions. When watching at Figure 3.13, then the default strategy executes experiments for base positions with green squares, red dots, and white dots. The capability map strategy knows that white dots will lead to failure and therefore only executes experiments for base positions with green squares and red dots. The exploration strategy based on human data performs exploration by randomly sampling base positions within isobars around human manipulation places, as described above. The result is that the capability map approach is clearly superior to the default strategy. The exploration strategy based on human biasing is another improvement, but the difference to the capability map approach is minor.

The exploration algorithm based on human biasing is quite simple and has some disadvantages. For instance, it could not discover multi-modal distributions of successful examples if failures lie in between. However, if the goal is to start acting as soon as possible, and to develop a satisfyingly good Generalized Success Model as fast as possible, then the exploration strategy based on human biasing will speed up the exploration process.

3.6 Learning GSMs for Different Objects

Until now this chapter explained how to learn a model of successful manipulation places for grasping cups. In everyday manipulation a robot will face a multitude of different objects with different shapes and sizes. It is infeasible to learn Generalized Success Models for all kinds of objects. In this section we will show how to address this problem by learning Generalized Success Models for *grasps* instead of for learning GSMs for *objects*. A grasp is determined by

- a grasp point
- an approach vector

A grasp point is a 3D position in space where the target object can be grasped in a stable manner. An approach vector specifies from which direction the gripper should approach the grasp point. The grasp point for a cup, for example, is located at its handle and the approach vector points from the side towards the handle as can be seen in Figure 3.3. Now we will present a Generalized Success Model for grasping objects by approaching them from top to bottom. Further research by Maldonado et al. (2010) found out that most objects can be grasped by either approaching them from the side or from the top.

3.6.1 Grasping Different Objects

If robots act in everyday environments, they have to face a broad variety of different objects. It is not feasible to learn a specific Generalized Success Model for every object. Nevertheless, the goal is to use the ARPLACE framework for computing manipulation places for all kinds of objects. A solution to this dilemma is to learn places for performing different *grasps* instead of learning how to grasp different *objects*. The idea is that there are significantly less possibilities to grasp an object, than there are objects. A grasp primarily consists of a *grasp point* and an *approach vector* that specifies the direction from where the gripper approaches the grasp point. Consider a plate, for example. The grasp point is usually somewhere on the plate's edge, and the approach vector faces sideways into the plate as can be seen in Figure 3.19. This is very similar to the grasp point and approach vector of cups.

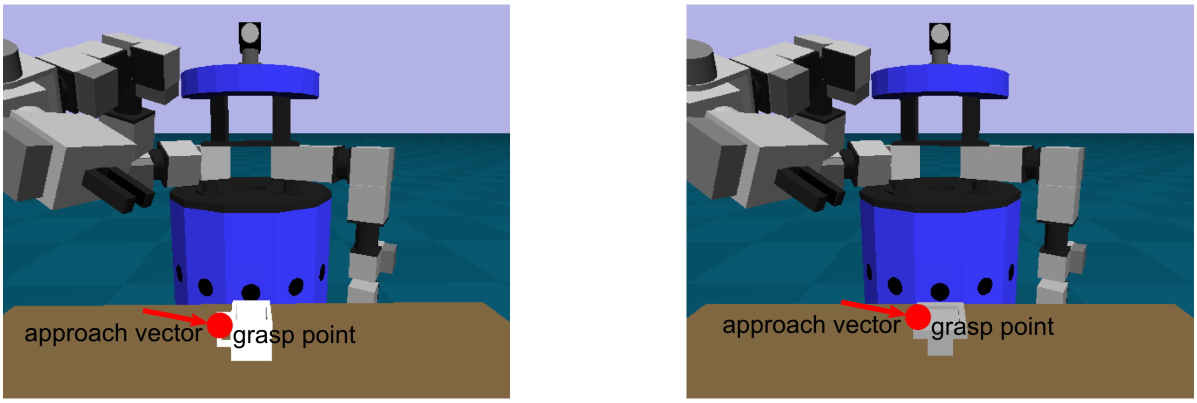


FIGURE 3.19 Grasp point (red dot) and approach vector (red arrow) for grasping a cup (left image) and a plate (right image). The grasp points and approach vectors are nearly identical. The major difference is that the gripper orientation is vertical for grasping cups and horizontal for grasping plates.

A reaching motion for grasping plates is shown in Figure 3.20. The reaching motions that the vector field controller generates in order to grasp cups and plates are very similar, as can be seen when comparing Figure 3.20 to Figure 3.3. The major difference is that the robot will have to rotate its gripper by 90°because the handle of a plate is horizontal, while the handle of a cup is vertical. When considering that the sixth arm joint (joint *5* in Figure A.3 because joint labeling starts with '0') is for rotating the gripper, it is obvious that grasping a plate is the same as grasping a cup for the first five joints, while the sixth joint is turned by additional 90°. Please note that the gripper for grasping the plate is oriented in a way that the dark gray box, that indicates the laser scanner that is mounted on the gripper, is on top in order not to collide with the table.

Cutlery or writing utensils are best grasped in the center with the gripper approaching from top to bottom. The reaching motion for grasping a knife together with the grasp point and approach vector is depicted in Figure 3.21.

Please note that the knife is positioned on a white sponge for safety reasons. It is possible that our vision system erroneously estimates the target object's pose to be beneath the table. In this case the robot manipulator would consequently try to grasp the target object at the

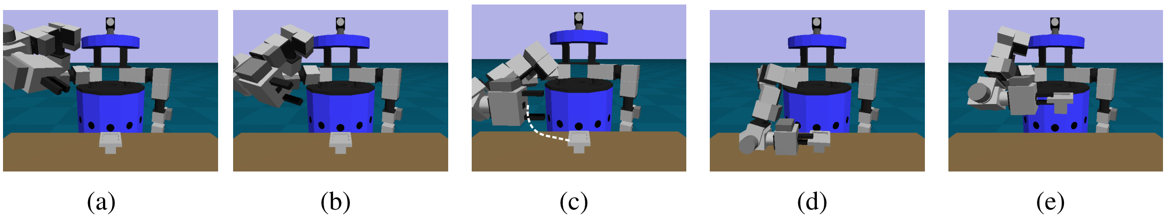


FIGURE 3.20 Reaching motion for grasping plates. a) Robot reached target base position. b) Robot opened gripper and started to rotate the arm. c) Robot gripper now faces to the left with the gap between the gripper being horizontal. d) Endeffector positioned around plate and gripper closed. e) Robot successfully lifted the plate.

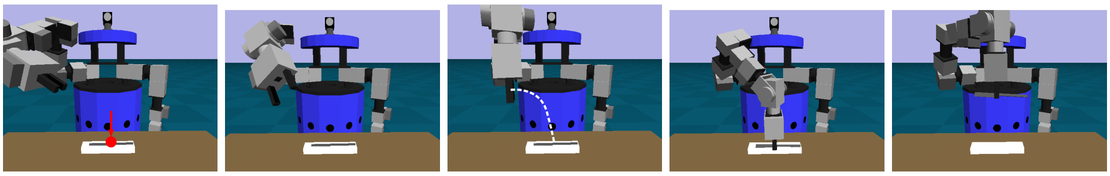


FIGURE 3.21 Reaching motion for grasping knifes.

estimated pose and crash into the table. Because our Powercube manipulator does not have force control this can lead to severe damage. That is why the height of the table is coded into the vector field controller. In the vector field, the table plate applies a huge repelling force in regions that are close to the table plate (approximately in the 2cm above the table plate). When an object is small and located directly on the table, then it is within the region of repelling force, and manipulator oscillation might occur. The manipulator tries to approach the object and is pushed away from the table plate by the repelling force. When the manipulator's distance to the table plate is big enough so that the attracting force of the target object exceeds the repelling force of the table, then the manipulator approaches the table plate again. Although this behavior is not desired, it prevents the manipulator from crashing into the table and has proven to be stable. The only drawback is that no objects can be grasped that are less than 2cm above the table plate, and the reason why we put the knife on a sponge that is 3cm high.

Finally, there are objects that can be grasped in multiple ways. A glass for example is usually grasped from top to bottom at its opening, but it is possible to grasp it from the side at its body. The default reaching motion for grasping a glass from top to bottom is depicted in Figure 3.22 together with grasp points and approach vectors.

Several grasp planners were developed that compute grasp points that lead to physically stable grasps. Miller and Allen (2000), for example, implemented GraspIt! that is publicly available. In order to find stable grasps like form and force closure grasps, a heuristic has to be optimized that captures grasp stability (Miller and Allen, 1999). Research done by other researchers of our group developed a system for model-free grasping. Maldonado et al. (2010)

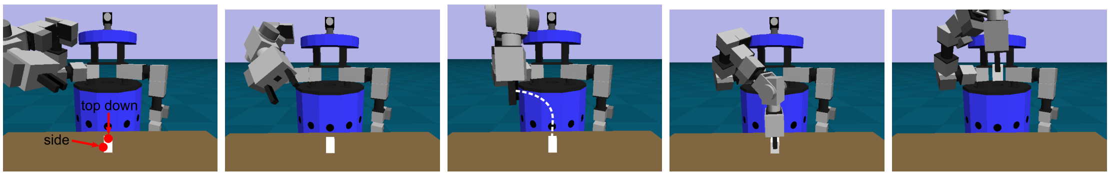


FIGURE 3.22 Reaching motion for grasping glasses.

describe the system that enables a robot to grasp unknown objects. A vision system uses a time-of-flight camera that is mounted on a pan-tilt unit in order to find the object's center and an approximation of its shape. The grasp pose is optimized based on gaussian point distributions, which ensures that the endeffector can grasp the target object in a physically stable way. It was found that grasps from the side and from the top are the most important grasps. "The result is intuitive: if an object is placed upright on a table, we only have to evaluate if we better grasp from the top or from the side. This does not hold for any inclined objects or any objects placed on a ramp, but it is valid for most household items and all our test objects" (Maldonado et al., 2010). The system was tested with the household items that can be seen in Figure 3.23.

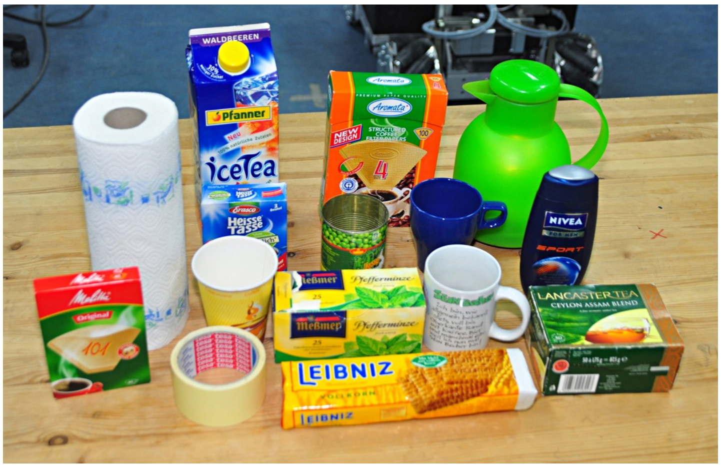


FIGURE 3.23 15 household items that were used to evaluate model-free grasping

In an evaluation on a real robot 48 out of 59 grasps succeeded, which leads to an overall success probability of 80%. The result is even stronger when it is considered that the Nivea shower gel was particularly difficult to grasp. As a result it was tested the most with eight grasps instead of three to four grasps for the other objects. Seven out of eight attempts for grasping the Nivea shower gel failed. Overall, the evaluation indicates that most household objects can be grasped from the side or from the top. Based on these results, we decided to learn Generalized Success Models for these two grasps.

3.6.2 A Generalized Success Model for Grasping from the Top

In this section we present a Generalized Success Model for grasping from the top. This section visualizes important results in the data acquisition and model learning phase and shows how grasping from the top differs from grasping from the side. In order to learn a Generalized Success Model for grasping objects from the top, we chose a glass as target object that was positioned at the following poses with respect to the relative feature space.

$$\Delta x_{\mathrm{obj}} \in \{0.07\mathrm{m}, 0.17\mathrm{m}, 0.27\mathrm{m}, 0.37\mathrm{m}, 0.47\mathrm{m}, 0.57\mathrm{m}, 0.67\mathrm{m}\}$$

$$\Delta \psi_{\mathrm{obj}} \in \{210°, 240°, 270°, 300°, 330°\}$$

This results in an overall of 35 object poses. The robot has to grasp the object from 693 different base positions that are uniformly distributed in a large rectangular area.

$$\Delta x_{\mathrm{rob}} \in \{\text{-}1.13\mathrm{m}, \text{-}1.08\mathrm{m}, \text{-}1.03\mathrm{m}, \text{-}0.98\mathrm{m},\ .., \text{-}0.18\mathrm{m}, \text{-}0.13\mathrm{m}\}\ ;$$

$$\Delta y_{\mathrm{rob}} \in \{\text{-}0.60\mathrm{m}, \text{-}0.55\mathrm{m}, \text{-}0.50\mathrm{m}, \text{-}0.45\mathrm{m},\ .., 0.95\mathrm{m}, 1.00\mathrm{m}\}\ ;$$

$$\Delta \psi_{\mathrm{rob}} = 0°\ ;$$

Figure 3.24 depicts all 24.255 experiments. Only a small portion had to be executed, because the capability map enabled us to label experiments as failures, that did not have a theoretical chance of succeeding (indicated by white circles). Please note, that although the object that is plotted seems to be a cup, the experiments were performed with a glass. The problem is that plotting a round glass makes it impossible to visualize its orientation. That is why a "handle" was included for plotting.

The data shows interesting properties. The orientation of the target object has no significant impact on manipulation places for successfully grasping from the top. This can be seen, when watching at each column of plots. A column of plots depicts configurations where the distance of the target object to the table edge remains static, while its orientation changes. All images in a column of plots have approximately the same distribution of successful grasps (indicated by green squares), and therefore the classification boundaries are very similar. The plots on the bottom below a black arrow depict all five classification boundaries from above in a single plot. The bottom left plot for example (shown in greater detail in the left image of Figure 3.25) depicts all classification boundaries for plots with $\Delta x_{\mathrm{obj}} = 0.07\mathrm{m}$. It can be seen that the classification boundaries mostly overlap. As a result, the classification boundaries' alignment can hardly be seen although it is plotted. This is a difference to grasping objects from the side,

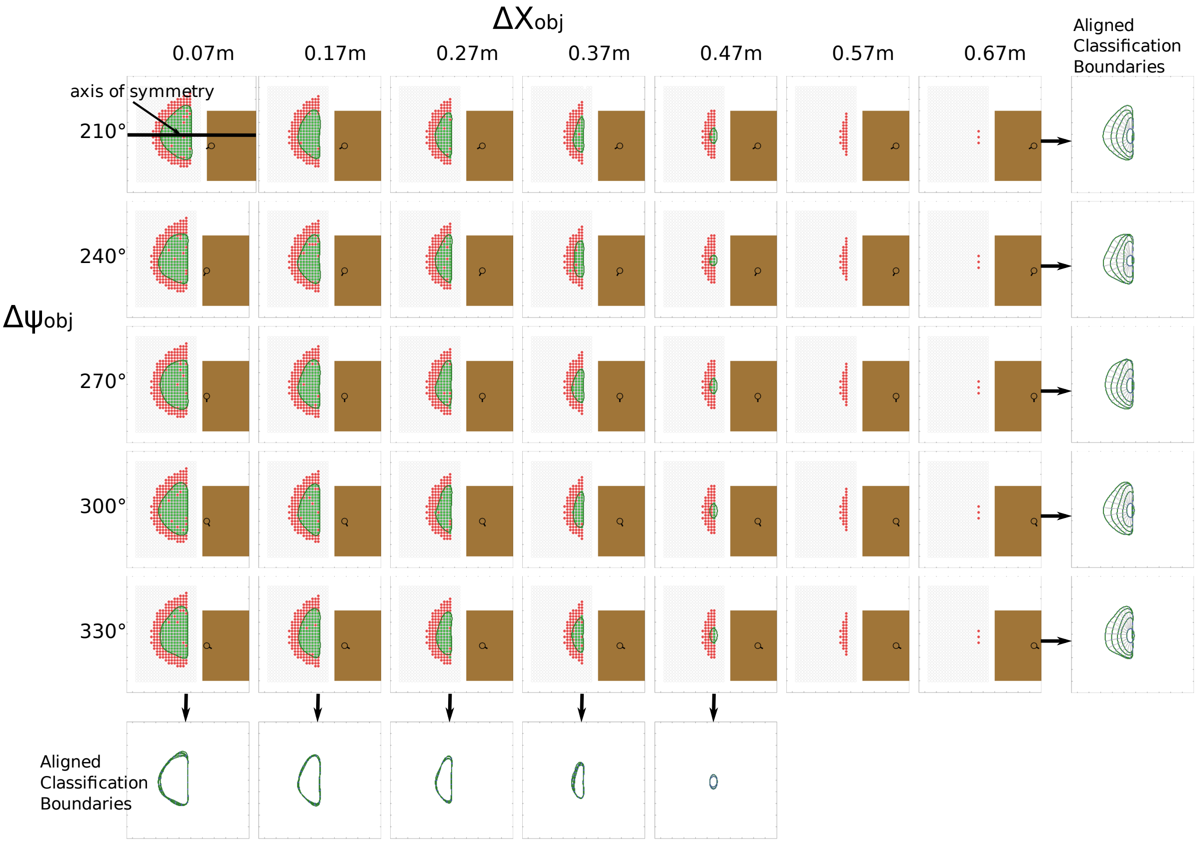


FIGURE 3.24 Gathered training data and learned classification boundaries for grasping objects from the top. Every subplot shows the training data that corresponds to the pose that is visualized with the black glass. The 'handle' visualizes the orientation of the glass. The black line in the top left plot indicates the axis of symmetry for the classification boundary. The classification boundaries for the data of a row or column of plots and their aligned classification boundaries are shown in special plots on the right and at the bottom.

where the orientation of the object had a significant impact on the place from where successful grasping actions can be performed.

Another effect of the fact that object orientation does not matter is that all classification boundaries are almost symmetrical. The black line in the top left plot of Figure 3.24 visualizes the axis of symmetry. The reason for this is that the robot does not have to grasp around an object, which was the case for grasping from the side. When grasping from the top the robot prefers to position itself in a way that its shoulder is approximately in front of the object. This means that the axis of symmetry is shifted slightly to the left of the object when grasping with the right arm.

It can be seen that the region for successful grasping gets smaller as the distance of the object to the table edge gets bigger. This property was already observed for grasping from

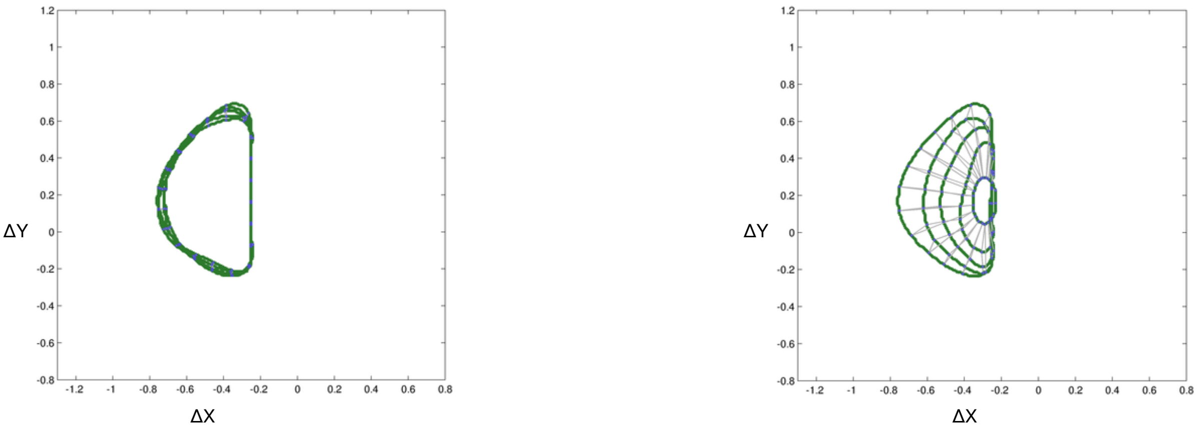


FIGURE 3.25 Left: Aligned classification boundaries for $\Delta x_{\text{obj}} = 0.07\text{m}$ and $\Delta\psi_{\text{obj}} \in \{210°, 240°, 270°, 300°, 330°\}$. Right: Aligned classification boundaries for $\Delta\psi_{\text{obj}} = 210°$ and $\Delta x_{\text{obj}} \in \{0.07\text{m}, 0.17\text{m}, 0.27\text{m}, 0.37\text{m}, 0.47\text{m}, 0.57\text{m}, 0.67\text{m}\}$. There are only five classification boundaries, because the object is not reachable from $\Delta x_{\text{obj}} \in \{0.57\text{m}, 0.67\text{m}\}$.

the side (compare Figure 3.13). The classification boundary approaches the table edge as the object moves further away from the table edge. While its shape remains relatively constant for object poses near the table edge, it changes when the object's distance to the table edge exceeds a certain threshold. The object cannot be grasped any more when the distance to the table edge is 0.57m or higher. Therefore, no classification boundaries can be computed for $\Delta x_{\text{obj}} \in \{0.57\text{m}, 0.67\text{m}\}$.

The plots on the right of a black arrow in Figure 3.24 depict the classification boundaries from the corresponding row of plots. The top right plot for example (shown in greater detail in in the right image of Figure 3.25) depicts all classification boundaries for plots with a object orientation of $\Delta\psi_{\text{obj}} = 210°$. It can be seen that the shape of the five classification boundaries are approximately the same, while their size differs. The right border is straight because the table limits the robot's base position in this direction. The left border is circular with the size depending on the distance of the object to the table edge. The alignment of the points on the classification boundaries can be seen clearly.

Figure 3.26 depicts the first three deformation modes of the point distribution model for grasping from the top. It can be seen that the first deformation mode represents the boundaries stretching to the back, that is dependent on the object's distance to the table edge. The other two deformation modes add only little additional deformation information because the solid bright green lines are nearly identical. All three deformation modes contain 99.3% of the deformation energy, while the first two deformation modes contain 99.0% of the deformation energy, and the first deformation mode already contains 94.3% of the deformation energy. Only storing the first deformation mode therefore results in a very compact, yet precise model of manipulation place for grasping from the top.

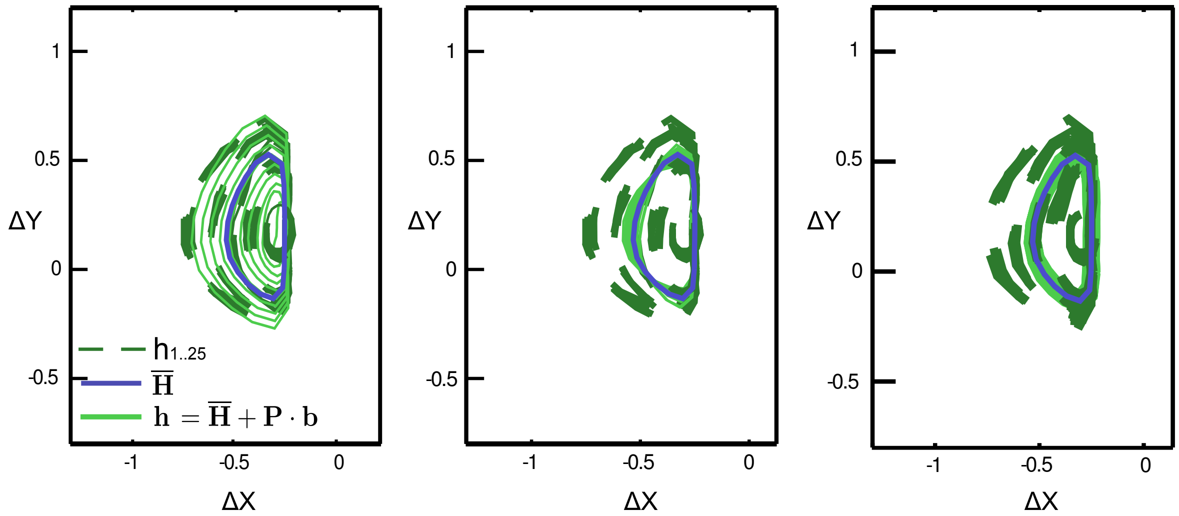


FIGURE 3.26 The first three deformation modes of the point distribution model for grasping from the top. The dashed green lines are the classification boundaries, as seen in Figure 3.24. The blue line is the mean of the deformation mode, and the solid bright green line are reconstructed classification boundaries for the corresponding deformation mode.

The fact that the target object's orientation has only minor impact on manipulation places opens a possibility to reduce the required number of experiments. The runtime for learning a Generalized Success Model for the complete training set with 24.255 experiments is presented in the second column of Figure 3.27. Additionally, we learned a Generalized Success Model where all distances to the table edge were considered as before (therefore the values of Δx_{obj} were as follows: $\Delta x_{\text{obj}} \in \{0.07m, 0.17m, 0.27m, 0.37m, 0.47m, 0.57m, 0.67m\}$), but this time we deleted all experiments that were not performed with an object orientation of $\Delta\psi_{\text{obj}} = 270°$. A reduced training set with 4.851 experiments remained. The Generalized Success Model that was learned from the reduced training set is almost identical to the model that was learned for the complete training set. The runtime for learning a Generalized Success Model for the complete and the reduced training set is presented in Figure 3.27.

When studying the code, linear computational complexity could be assumed for all computational steps with the exception of aligning classification boundaries is linear. The results shown above support this assumption. Computational complexity of aligning classification boundaries, however, is quadratic. 64.000.000 distances between landmarks had to be computed for the complete training set, while 2.560.000 distances had to be computed for the

Computational Step	**Time** (in seconds)	**Time** (in seconds)
1. Learn Classification Boundaries	4.3	0.8
2. Align Classification Boundaries	71.1	2.8
3. Learn Point Distribution Model	24.4	5.5
4. Relation to Task-Relevant Parameters	0.03	0.003
Overall	**99.8**	**9.1**

FIGURE 3.27 Runtime for learning a Generalized Success Model with the complete training set (2nd column) and the reduced training set (3rd column).

reduced training set.

CHAPTER 4

Action-Related Places

On an implementation level, an Action-Related Place (ARPLACE) discretizes 2D space into grid cells that are unambiguously identified by their indexes along the x-axis and y-axis. A probability value is assigned to every grid cell that captures the predicted probability of successfully grasping the target object when the grasping action is performed from within this grid cell. Therefore, given a robot, a target object, and an environmental context, an ARPLACE provides a mapping from grid cells to grasp success probability p:

$$f(x, y) : \mathbb{Z} \times \mathbb{Z} \rightarrow p \qquad ; \;\; p \in [0, .., 1]$$

The optimal manipulation place is the center of the grid cell that maximizes this function. The proposed manipulation place is therefore the one that maximizes the chance of successful grasping. An exemplary ARPLACE is depicted in Figure 4.1.

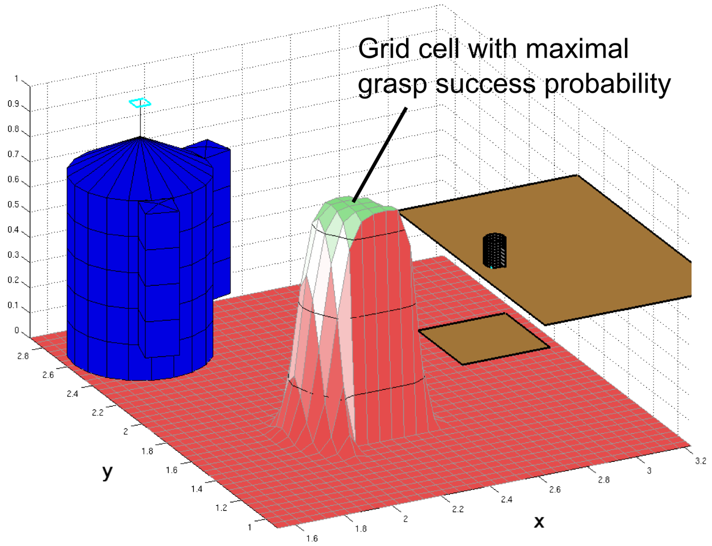


FIGURE 4.1 The robot, a chair, a table, and an ARPLACE for grasping the cup. A hole is bumped into the ARPLACE because the chair blocks several promising grid cells.

On a symbolical level we define ARPLACEs as the set of pairs $\langle$ *pos, p* $\rangle$, where *pos* is a robot pose relative to the table, and *p* is the predicted grasp success probability which we request to be higher than the threshold value θ:

$$
\begin{aligned}
&\text{ARPLACE (pickup(o))} := \{\ \langle \text{ pos, p } \rangle \mid \\
&\qquad\qquad \text{p} = \text{P(successful(ev)} \mid \text{occurs(ev,t), holds(at(rob,pos),t),} \\
&\qquad\qquad\qquad\qquad \text{event-type(ev,pickup), object-acted-on(ev,o))} \\
&\qquad\qquad \geq \theta\ \}
\end{aligned}
$$

In the last chapter we showed how to learn Generalized Success Models which are very compact models of grasp success probability for manipulation places. In this chapter we explain how Generalized Success Models are used online in order to compute ARPLACEs that are based on grasp success probability. The algorithm that computes grasp success probability explicitly takes the entropy of an environmental state into account. Within our robot system, the entropy of a state is measured by its perception systems, namely the localization system and the vision-based object detection system. The entropy is then represented in covariance matrices that represent the uncertainty of the robot into its estimations of the robot's base position (for the localization system) and of the object's pose (for the vision-based object detection system). Taking entropy into account by representing it through state estimation uncertainties leads to recommended manipulation places that are more robust.

In order to compute grasp success probability the robot probabilistically estimates its own base position and the poses of target objects, and uses these state estimations as input for querying the Generalized Success Model. Instead of committing to a specific manipulation place in advance, ARPLACE gets updated as new sensor data comes in. This is possible because ARPLACEs can be computed very fast and enables least commitment planning.

The remainder of this chapter is structured as follows. Related work is presented in Section 4.1. In Section 4.2 we show how the robot computes ARPLACEs that are based on grasp success probability. Section 4.3 presents an evaluation that analyzes the impact of all task-relevant parameters on the resulting ARPLACE probability distribution. Section 4.4 describes that ARPLACEs provide valuable information for high level planning systems, and section 4.5 presents results from the simulated B21r robot.

4.1 Related Work

Okada et al. (2006) denote a good base placement for grasping a 'spot'. Different spots are hand-coded for different tasks, such as manipulating a faucet, a cupboard, and a trashcan.

These symbolic representations of place are then used by a LISP-based motion planner to perform tool manipulation behavior. ARPLACE extends the concept of a spot by learning it autonomously, grounding it in observed behavior, and providing a probabilistic representation of place.

Berenson et al. (2008) address the issue of finding optimal start and goal configurations for manipulating objects in pick-and-place operations. They explicitly take the placement of the mobile base into account. As they are interested in the optimal start and goal configurations instead of a probabilistic representation, this approach does not enable least-commitment planning.

Diankov et al. (2008) use a model of the reachable workspace of the robot arm to decide where the robot may stand in order to grasp an object and to guide the search. However, uncertainties in robot base position or object poses are not considered and thus can not be compensated.

Zacharias et al. (2009) examine the question of how robot manipulators and the robot's locomotion system can complement each other in order to successfully perform a manipulation action in household tasks. For a given task such as opening a closet, a 3D manipulator trajectory is generated by searching the manipulator's capability map using correlation, and validating candidate trajectories. The robot's base pose is determined by aligning the end point of the 3D trajectory with the target object and choosing the robot's base in a way that the manipulator is aligned with the start point of the trajectory. The difference between this approach and ours is the importance that is attributed to finding good manipulation places. Zacharias et al. (2009) start with computing a 3D manipulator trajectory and subsequently choose the robot's base pose in a way that the manipulator trajectory can be executed. ARPLACEs start with finding optimal base positions by taking state estimation uncertainties into account. However, ARPLACEs do not create manipulator trajectories, but delegate this task to the manipulation system.

4.2 Computing Action-Related Places

In this section, we describe the online computation ARPLACEs. Figure 4.2 serves as an outline of this section.

The image depicts that the online part of the ARPLACE framework expects three inputs. First, the Generalized Success Model that was learned by the offline part of the ARPLACE framework, as explained in chapter 3. Second, the robot's base position that is estimated by the localization system. And third, the target object's pose that is estimated by the vision system.

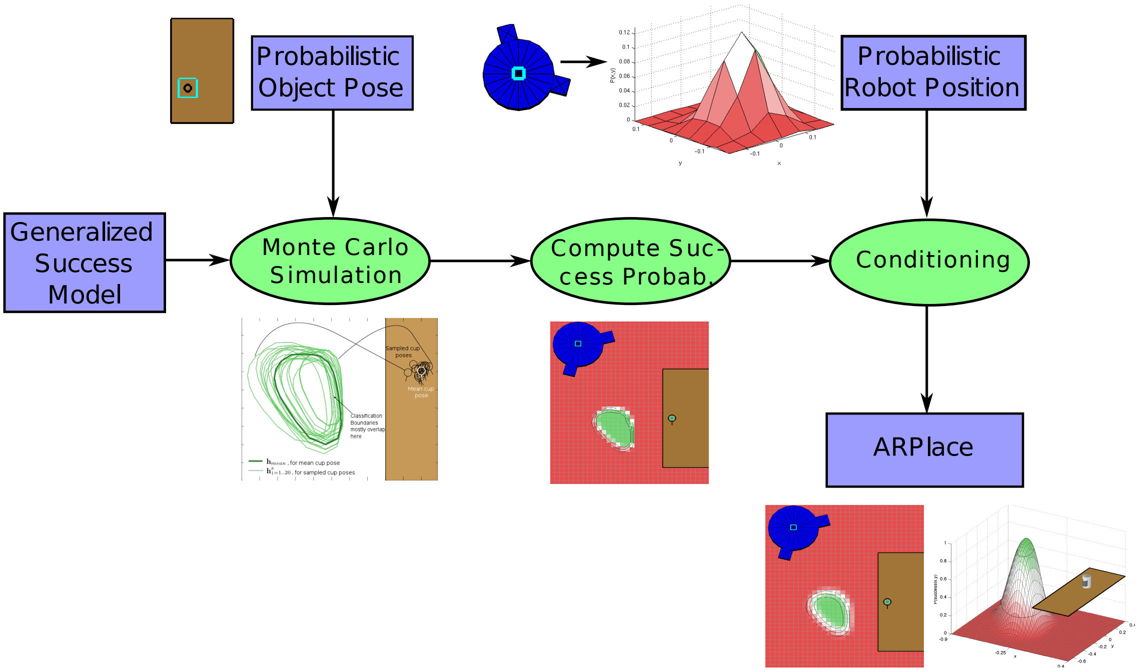

Probabilistic Object Pose
Probabilistic Robot Position
Generalized Success Model
Monte Carlo Simulation
Compute Success Probab.
Conditioning
ARPlace
Sampled cup poses
Mean cup pose
Classification Boundaries mostly overlap here
$\mathbf{h}_{mean}$, for mean cup pose
$\mathbf{h}^s_{i=1..20}$, for sampled cup poses
P(x,y)
x
y

FIGURE 4.2 Visual outline of the chapter. Green ellipses represent computational steps for computing ARPLACEs. Blue rectangles represent data structures that are used in the computations. Images near blue rectangles are exemplary visualizations of the corresponding data structure.

Careful readers will remember that Figure 2.7 depicted that a high level planner transmits parameters of task context. However, this chapter deals with ARPLACE distributions that are based on grasp success probability. Parameters of task context such as how important it is to perform the task as quickly as possible are only required if the more general utility-based ARPLACEs are computed. We will adress this topic in chapter 6.

Because the localization system estimates the robot's base position with respect to the world frame, and the vision system estimates object poses with respect to the camera's coordinate frame, we have to transform base positions and object poses to the relative feature space. How this is done is described in section 4.2.1. Section 4.2.2 explains the algorithms (green ellipses in Figure 4.2) that are used to compute ARPLACE. First, it is shown how to compute ARPLACEs when there is ground truth data in section 4.2.2.1. Sections 4.2.2.2 and 4.2.2.3 describe how the ARPLACE framework takes state estimation uncertainty into account.

4.2.1 From Robot Coordinate Systems to the Relative Feature Space

Our robot uses a variety of coordinate systems. The localization system for example estimates the robot's base position with respect to the world frame. The target object's pose is estimated with respect to the camera of the vision system. In the Generalized Success Model however, robot positions and object poses are encoded with respect to the relative feature space, that was described in secion 3.3.1. The reason to use the relative feature space was that a single Generalized Success Model is enough to

- capture manipulation places for grasping from any table side
- handle translations and rotations of supporting tables

When we want to query the Generalized Success Model, we have to compute the matrix ${}^{\Delta}T_O$ that describes the target object's pose with respect to the relative feature space. In the following we present the required coordinate frames and how they are transformed in order to obtain ${}^{\Delta}T_O$. The coordinate frames depicted in Figure 4.3 are involved in the transformation: the world frame F_W, the robot frame F_R that is centered in the robot's base position at the floor, the frame of the pan-tilt unit where the camera is mounted F_{PT}, the camera frame F_C that is centered in the camera's sensor chip, the table frame F_T that is centered in the middle top of the table, and the relative feature space F_Δ.

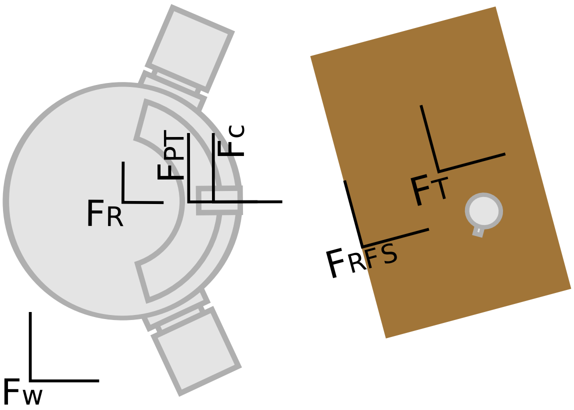


FIGURE 4.3 Relevant coordinate frames for computing the target object's pose with respect to the relative feature space.

The first step is to compute the object's pose with respect to the world frame ${}^{W}T_O$. Because the robot is not able to get this information directly it has to perform the following coordinate transformations

$$ {}^{W}T_O = {}^{W}T_R * {}^{R}T_{PT} * {}^{PT}T_C * {}^{C}T_O \tag{4.1} $$

All four homogenous transformation matrices on the right side are known or can be inferred by the robot as we will describe now. The base position of the robot with respect to the world frame is estimated continually by the robot's localization system. The estimation includes the values $\langle x_{\text{rob}}, y_{\text{rob}}, \psi_{\text{rob}}\rangle$ for the mean position that can be transformed into a 4x4 homogenous transformation matrix ${}^{W}T_{R}$. In general, 4x4 homogenous transformation matrices in 3D space (HT_{3D}) have six degrees of freedom $\langle x, y, z, \phi, \theta, \psi\rangle$ and look as follows

$$HT_{3D} = \begin{pmatrix} rot_{11} & rot_{12} & rot_{13} & x \\ rot_{21} & rot_{22} & rot_{23} & y \\ rot_{31} & rot_{32} & rot_{33} & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The top left 3x3 matrix represents the rotation between coordinate frames as specified by the roll, pitch, and yaw angle $\langle \phi, \theta, \psi\rangle$. The top right 3x1 matrix represents the translation as specified by $\langle x, y, z\rangle$. The same is done with the covariance values $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}}, \Delta\psi_{\text{rob}}\rangle$ in order to compute the covariance matrix ${}^{W}T_{R_{cov}}$ that represents the uncertainty of the localization system into the estimated mean position. In general, covariance matrix for a 3D pose (Cov_{3D}) is a 6x6 matrix and looks as follows

$$Cov_{3D} = \begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) & Cov(x,\phi) & Cov(x,\theta) & Cov(x,\psi) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) & Cov(a,\phi) & Cov(a,\theta) & Cov(a,\psi) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) & Cov(z,\phi) & Cov(z,\theta) & Cov(z,\psi) \\ Cov(\phi,x) & Cov(\phi,y) & Cov(\phi,z) & Cov(\phi,\phi) & Cov(\phi,\theta) & Cov(\phi,\psi) \\ Cov(\theta,x) & Cov(\theta,y) & Cov(\theta,z) & Cov(\theta,\phi) & Cov(\theta,\theta) & Cov(\theta,\psi) \\ Cov(\psi,x) & Cov(\psi,y) & Cov(\psi,z) & Cov(\psi,\phi) & Cov(\psi,\theta) & Cov(\psi,\psi) \end{pmatrix}$$

The matrices ${}^{W}T_{R}$ and ${}^{W}T_{R_{cov}}$ however do not have full rank because we assume the robot to remain upright on planar ground. That is why z_{rob}, ϕ_{rob}, and θ_{rob} are not estimated by Player's AMCL localization system. ${}^{W}T_{R}$ is therefore a simplified version of HT_{3D}.

$$
{}^{W}T_{R} = \begin{pmatrix} \cos(\psi_{\text{rob}}) & -\sin(\psi_{\text{rob}}) & 0 & x_{\text{rob}} \\ \sin(\psi_{\text{rob}}) & \cos(\psi_{\text{rob}}) & 0 & y_{\text{rob}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

The localization system estimates the covariance values $Cov(x_{\text{rob}}, x_{\text{rob}})$ and $Cov(y_{\text{rob}}, y_{\text{rob}})$, so ${}^{W}T_{R_{cov}}$ is a simplified version of Cov_{3D}.

$$
{}^{W}T_{R_{cov}} = \begin{pmatrix} Cov(x_{\text{rob}}, x_{\text{rob}}) & 0 \\ 0 & Cov(y_{\text{rob}}, y_{\text{rob}}) \end{pmatrix}
$$

The standard deviations $\sigma(x_{\text{rob}}, x_{\text{rob}})$ and $\sigma(y_{\text{rob}}, y_{\text{rob}})$ are the positive square roots of the above covariance values $Cov(x_{\text{rob}}, x_{\text{rob}})$ and $Cov(x_{\text{rob}}, x_{\text{rob}})$.

$$
\sigma(x_{\text{rob}}, x_{\text{rob}}) = \sqrt{Cov(x_{\text{rob}}, x_{\text{rob}})} \tag{4.2}
$$

$$
\sigma(y_{\text{rob}}, y_{\text{rob}}) = \sqrt{Cov(y_{\text{rob}}, y_{\text{rob}})} \tag{4.3}
$$

Compared to covariances standard deviations have the advantage that they are measured in the same unit as the underlying parameter. In this case meters. A standard deviation of $\sigma(x_{\text{rob}}, x_{\text{rob}}) = 0.04\text{m}$ for example, indicates that the robot believes that its estimated mean position with respect to the x-axis is not further away than 0.04m from the true position with a probability of 68.3%, not further away than 0.08m with a probability of 95.4%, and not further away than 0.12m with a probability of 99.7%. In the following we will refer to $\sigma(x_{\text{rob}}, x_{\text{rob}})$ and $\sigma(y_{\text{rob}}, y_{\text{rob}})$ with the abbreviations $\sigma_{x_{\text{rob}}}$ and $\sigma_{y_{\text{rob}}}$.

The pan tilt unit is mounted tightly to the robot platform. As there is no slip, the pose of the pan tilt unit with respect to the robot's base ${}^{R}T_{PT}$ is fix and does not change during operation. The corresponding 6x6 covariance matrix ${}^{R}T_{PT_{cov}}$ is the zero matrix $0_{6,6}$. The distances and angular offsets from the B21r robot base to the pan tilt unit were carefully measured by hand. The acquired values were used in order to compute ${}^{R}T_{PT}$. We assume maximum errors of 1mm for the distance measurements along the x-, y-, and z-axis, and 2° for the yaw angle measurements. Roll and pitch angles are zero, as the pan tilt unit is mounted level with respect to the ground.

The pan tilt unit is panned and tilted by the robot in order to focus the camera to the region of interest. Therefore, the pose of the camera's sensor relative to the pan tilt unit ${}^{PT}T_C$ changes according to the current pan and tilt angles. This information can be obtained from the pan tilt unit's driver with high accuracy and is transformed into the homogenous transformation matrix ${}^{PT}T_C$. No uncertainty is measured by the pan tilt unit's driver, so we set the 6x6 covariance matrix ${}^{PT}T_{C_{cov}}$ to the zero matrix $0_{6,6}$.

The object's pose relative to the robot's camera is estimated continuously by a vision-based object detection system. The vision system matches CAD models for finding objects in an image and estimates their full 3D pose with all six degrees of freedom. The estimation includes a 4x4 homogenous transformation matrix ${}^{C}T_O$ with full rank for estimating the object's mean pose. The uncertainty into this mean pose is captured in a 6x6 covariance matrix ${}^{C}T_{O_{cov}}$ where the diagonal elements are estimated, and the other values are zero. In the following we will compute the standard deviations of the covariance values and refer to $\sigma(x_{\text{obj}}, x_{\text{obj}})$ as $\sigma_{x_{\text{obj}}}$, to $\sigma(y_{\text{obj}}, y_{\text{obj}})$ as $\sigma_{y_{\text{obj}}}$, to $\sigma(z_{\text{obj}}, z_{\text{obj}})$ as $\sigma_{z_{\text{obj}}}$, to $\sigma(\phi_{\text{obj}}, \phi_{\text{obj}})$ as $\sigma_{\phi_{\text{obj}}}$, to $\sigma(\theta_{\text{obj}}, \theta_{\text{obj}})$ as $\sigma_{\theta_{\text{obj}}}$, and to $\sigma(\psi_{\text{obj}}, \psi_{\text{obj}})$ as $\sigma_{\psi_{\text{obj}}}$.

When the target object is a cup, then typical values along the diagonal of ${}^{C}T_{O_{cov}}$ are $\sigma_{x_{\text{obj}}} = 0.04\text{m}$, $\sigma_{y_{\text{obj}}} = 0.04\text{m}$, $\sigma_{z_{\text{obj}}} = 0.07\text{m}$, $\sigma_{\phi_{\text{obj}}} = 0.06\text{rad}$, $\sigma_{\theta_{\text{obj}}} = 0.06\text{rad}$, and $\sigma_{\psi_{\text{obj}}} = 0.8\text{rad}$. The values indicate that the estimation of a cup's pose is quite accurate except for . The vision system has problems to detect the handle which is important for estimating yaw-orientation (ψ). For more information about the vision-based object detection system, please have a look at the publications of Klank et al. (2009a) and Klank et al. (2009b).

By using the above four homogenous transformation matrices, we can compute the object's mean pose with respect to the world frame ${}^{W}T_O$ as specified in formula 4.1.

$$ {}^{W}T_O = {}^{W}T_R \cdot {}^{R}T_{PT} \cdot {}^{PT}T_C \cdot {}^{C}T_O $$

Please note that the estimated pose of the localization system ${}^{W}T_R$ is critical, because errors are magnified by subsequent matrix multiplications. The uncertainty of the robot into the object's mean pose is represented by ${}^{W}T_{O_{cov}}$ and computed as follows

$$ {}^{W}T_{O_{cov}} = {}^{W}T_{R_{cov}} \cdot {}^{R}T_{PT_{cov}} \cdot {}^{PT}T_{C_{cov}} \cdot {}^{C}T_{O_{cov}} \tag{4.4} $$

The next step is to compute the origin of the relative feature space with respect to the world frame ${}^{W}T_{RFS}$. Therefore, the pose of the table with respect to the world frame ${}^{W}T_T$ is required. There are two possibilities to determine ${}^{W}T_T$. For static tables, all values can be directly derived from a map. In dynamic environments the vision system has to estimate the values. The vision system estimates the pose of the table relative to the camera ${}^{C}T_{T}$ in the same way as it estimates the pose of objects ${}^{C}T_{O}$. If the robot detected a table in the image, it can derive length, width, and height of the table from the matching CAD model. Subsequently, ${}^{W}T_{T}$ can be computed as follows

$$ {}^{W}T_{T} = {}^{W}T_{R} \cdot {}^{R}T_{PT} \cdot {}^{PT}T_{C} \cdot {}^{C}T_{T} \tag{4.5}$$

The next step is to find the origin of the relative feature space. Section 3.3.1 already described the computation when the closest table edge is known. This is the case for the offline part of the ARPLACE framework, because for model learning it is sufficient to perform manipulation from just one table edge. In the online part, however, the robot has to choose the most appropriate manipulation place among all table edges. Figure 4.4 depicts that there is one relative feature space for every table edge. This is intuitive, because the robot can perform manipulation actions from any table edge. And therefore a separate ARPLACE has to be computed for every table edge.

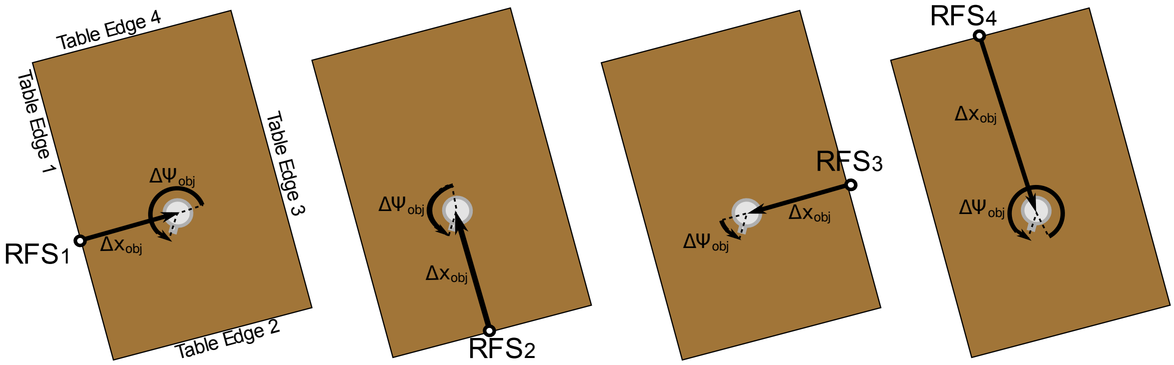


FIGURE 4.4 Because the robot can grasp the target object from any table edge, one ARPLACE has to be computed for every table edge. The relative feature space assures that a learned GSM is valid for all table edges. The picture depicts a table with four table edges and the origins of the relative feature spaces RFS_1, RFS_2, RFS_3, and RFS_4.

The following enumeration lists the computational steps that are required to derive the pose of the target object with respect to the relative feature space of table edge 1 (${}^{RFS_1}T_O$). We indicate parameters that are specified with respect to the relative feature space with a leading Δ. A more thorough explanation is presented in Section 3.3.1. All computational steps of the enumeration are illustrated in Figure 4.5.

1. Use ${}^{W}T_{T}$, table length, and table width to derive the vector of table edge 1 ($\overrightarrow{TE_1}$)

2. Use ${}^{W}T_{O}$ and $\overrightarrow{TE_1}$ to derive the normal of table edge 1 ($\overrightarrow{n}_{TE_1}$)

3. Use $\overrightarrow{TE_1}$ and $\overrightarrow{n}_{TE_1}$ to derive the origin of RFS_1 with respect to the world frame (${}^{W}T_{RFS_1}$). The x-axis of RFS_1 points from RFS_1 to the object and the y-axis of RFS_1 points along $\overrightarrow{TE_1}$.

4. Use ${}^{W}T_{RFS_1}$, $\overrightarrow{n}_{TE_1}$, and ${}^{W}T_O$ to derive the distance of the object to table edge 1. This is the task-relevant parameter Δx_{obj}

5. Use x-axis of ${}^{W}T_{RFS_1}$ and ${}^{W}T_O$ to derive the angular orientation of the object with respect to RFS_1. This is the task-relevant parameter $\Delta\psi_{\text{obj}}$

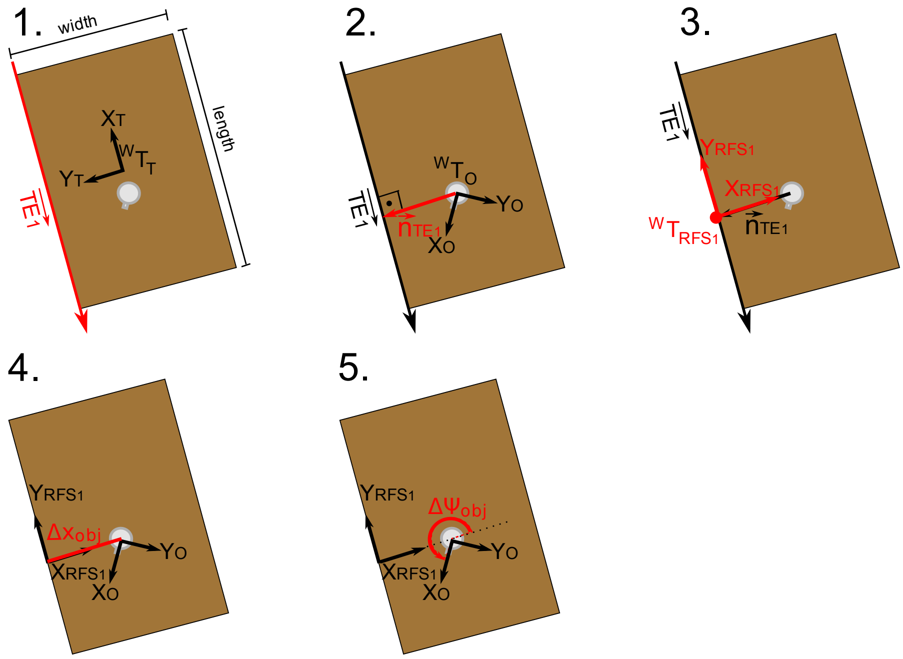


FIGURE 4.5 Each image visualizes a computational step from the enumeration above. Black elements are used as input. In the corresponding step red elements are the ones that are computed.

After performing the above steps, we get the required values Δx_{obj} and $\Delta\psi_{\text{obj}}$. The covariance values $\sigma_{x_{\text{obj}}}$ and $\sigma_{\psi_{\text{obj}}}$ are transformed accordingly in order to obtain $\sigma_{\Delta x_{\text{obj}}}$ and $\sigma_{\Delta\psi_{\text{obj}}}$. Please note that Δy_{obj} is 0.0m by definition. What does change is the origin of the relative feature space with respect to the table edge. So for different positions of the object, the origin of the relative feature space will be at different locations with respect to $\overrightarrow{TE_n}$. However, there is uncertainty into the object's position along the y-axis ($\sigma_{\Delta y_{\text{obj}}}$).

A Generalized Success Model that was learned for a certain table can be applied to tables with different table shapes, as long as step 1. for computing table edges is adapted accordingly. Figure 4.6 depicts a hexagonal table and a round table. It is straightforward to extend the presented method to hexagonal tables. One solution to handle round tables is to divide it into a polygon with several vertices. Each vertix can be seen as a table edge from where the robot can manipulate. For every table edge, however, an ARPLACE has to be computed. Dividing the circle into more vertices will lead to a more precise approximation of the table, but increases computational time.

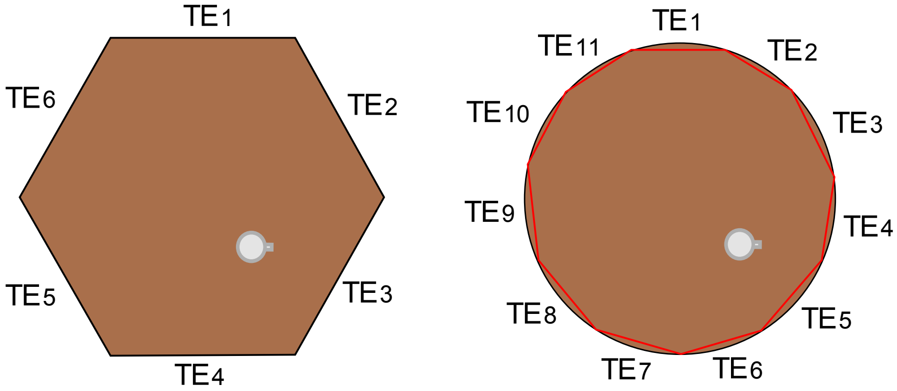


FIGURE 4.6 A GSM that was learned for a certain table can be applied to tables with different shapes. Left: Hexagonal table with six table edges. Right: Round table that is approximated by a polygon with eleven vertices (colored red). Each vertix can be treated like a table edge.

4.2.2 Querying the Generalized Success Model

In section 4.2.1 we described how the robot processes data from its localization and vision system in order to compute the homogenous transformation matrices ${}^{W}T_{O}$ and ${}^{W}T_{T}$, and how these matrices are used in order to estimate the target object's mean pose with respect to the relative feature space. The resulting variables Δx_{obj} and $\Delta\psi_{\text{obj}}$ are called task relevant parameters. Task-relevant parameters (the pose of the target object) specify the context of a manipulation action and the robot is not able to directly control them. We furthermore explained how to derive the covariance matrix ${}^{W}T_{O_{cov}}$ that describes the vision system's uncertainty into the estimated object pose. This uncertainty is specified with the parameters $\sigma_{\Delta x_{\text{obj}}}$ and $\sigma_{\Delta y_{\text{obj}}}$.

Moreover, section 4.2.1 showed how the localization system estimates the robot's base position ${}^{W}T_{R}$. The position of the robot with respect to the relative feature space is specified by $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle$. We call Δx_{rob} and Δy_{rob} controllable parameters, because the robot can control them directly by using its navigation system in order to move around. The uncertainty into the robot's estimated base position with respect to the world is represented by the covariance matrix ${}^{W}T_{R_{cov}}$ from which we derive $\sigma_{\Delta x_{\text{rob}}}$ and $\sigma_{\Delta y_{\text{rob}}}$.

4.2.2.1 Computing an Action-Related Place with Ground Truth Data

At the end of Section 3.4.3, we demonstrated how task relevant parameters are used to reconstruct a classification boundary $\mathbf{h}$. The Generalized Success Model predicts that whenever the robot moves to a base position $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle$ that is within the reconstructed classification boundary so that $\langle \Delta x_{\text{rob}}, \Delta y_{\text{rob}} \rangle \in \mathbf{h}$ and performs the manipulation action from there, then the manipulation action will succeed.

Figure 4.7 depicts a scenario where a cup is located on a table and the robot has ground truth data about the cup's pose. It furthermore depicts the reconstructed classification boundary and the manipulation places that the ARPLACE framework proposes for performing the manipulation action. Please note that the ARPLACE distribution that is presented here is boolean. A manipulation place is either considered to be promising (colored in bright green in Figure 4.7), or it is not (all other places). Due to sensor noise and other factors that influence state estimation, the task relevant parameters are usually not known exactly, and uncertainty into pose estimations must be modeled.

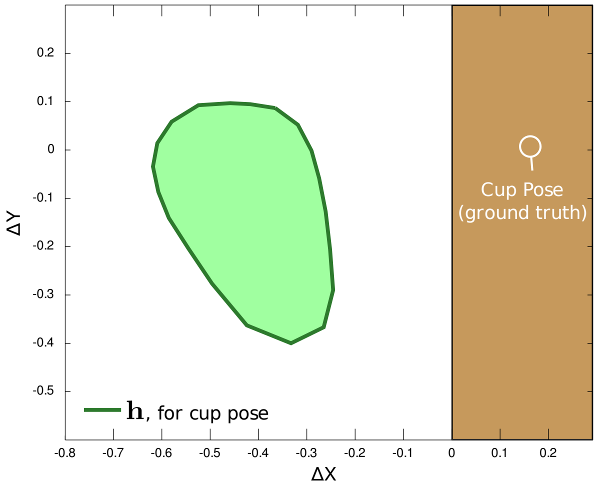


FIGURE 4.7 Target object is a white cup. The dark green line is the classification boundary that was reconstructed for the cup's pose. The GSM proposes to perform manipulation from places within the classification boundary (colored in bright green).

4.2.2.2 Uncertainty in Object Pose

Because of uncertainties of the vision system into the object's pose, it does not suffice to compute only one classification boundary given the most probable object pose. This might lead to failure if the object is not at the position where the robot expects it to be. Our solution to this problem is not to reconstruct one classification boundary for the object's estimated mean pose, but to reconstruct many classification boundaries by sampling object poses from the gaussian distribution that is defined by the estimated mean pose and the covariance matrix. We perform a Monte Carlo simulation by sampling object poses from this distribution, and reconstruct a classification boundary for each sample.

We sample 100 object poses given its mean pose $\langle \Delta x_{\mathrm{obj}}, \Delta\psi_{\mathrm{obj}} \rangle$ and standard deviations $\langle \sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{y^{\mathrm{tab}}_{\mathrm{obj}}}, \sigma_{\Delta\psi_{\mathrm{obj}}} \rangle$. The sampling step results in 100 pose samples t^s_i, where $t^s_i = \langle \Delta x^s_{\mathrm{obj}}, \Delta\psi^s_{\mathrm{obj}} \rangle$ and i is the index of the sample ranging from 1 to 100. For every pose sample, the corresponding classification boundary h^s_i is reconstructed. Figure 4.8 depicts the mean pose of a cup, 20 out of 100 cup poses that were sampled, and 20 reconstructed classification boundaries that correspond to the 20 sampled cup poses. Because the target object is a cup, the Generalized Success Model for grasping from the side was used for reconstructing the classification boundaries. The task relevant parameters used in this example are $\langle \Delta x_{\mathrm{obj}}, \Delta\psi_{\mathrm{obj}} \rangle = \langle 0.2\mathrm{m}, -\frac{\pi}{2} \rangle$, and $\langle \sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}}, \sigma_{\Delta\psi_{\mathrm{obj}}} \rangle = \langle 0.03\mathrm{m}, 0.03\mathrm{m}, 0.3\mathrm{rad} \rangle$.

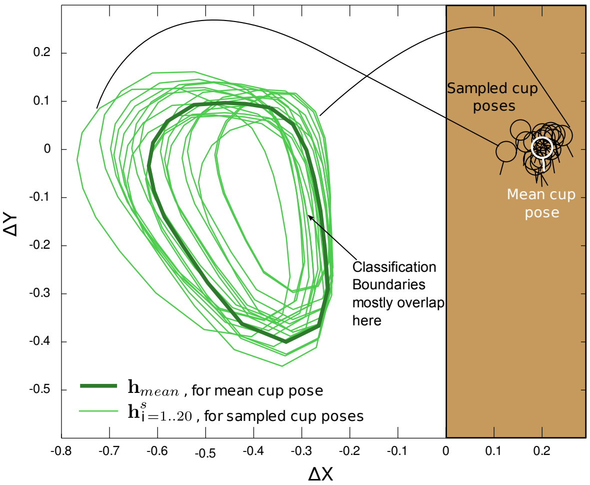


FIGURE 4.8 Monte-Carlo simulation of object poses. The white cup visualizes the robot's estimation of the cup's mean pose. The black cups are 20 out of 100 cup poses that were sampled (t^s_i). The dark green polygon is the reconstructed classification boundary for the mean cup pose. The light green polygons are the reconstructed classification boundaries (h^s_i). The black lines connect two sampled cup poses with their reconstructed classification boundary.

After having reconstructed the classification boundaries, we discretize space. The left image of Figure 4.9 depicts the 20 reconstructed classification boundaries from Figure 4.8 together with a grid where each grid cell measures 2.5x2.5cm. For every grid cell we compute by how many classification boundaries it is surrounded. Three grid cells are colored. The red grid cell for example is surrounded by 0 classification boundaries, the orange grid cell is surrounded by 7 classification boundaries, and the green grid cell is surrounded by all 20 classification boundaries. This means that when the robot moves to the green grid cell it will be able to successfully perform the grasp for any sampled object pose. It will never be successful when moving to the red grid cell. It is obvious that classification boundaries will be more spread out when the uncertainty about the object's pose increases. If the uncertainty is above a certain threshold, there will be no grid cell any more that is included in all classification boundaries.

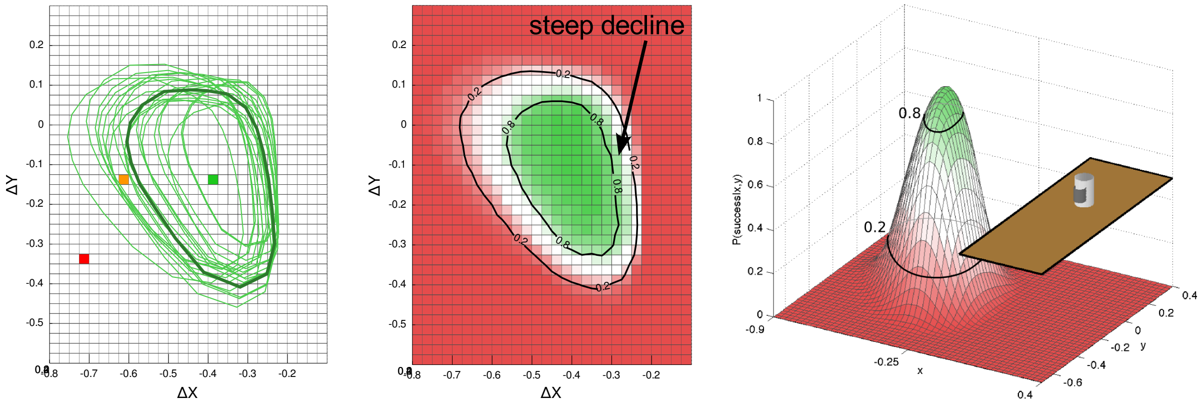


FIGURE 4.9 Computing an ARPLACE probability distribution. Left: The red, orange, and green grid cells are included in 0, 7, and 20 classification boundaries. Center: A color is assigned to each grid cell according its grasp success probability. Isobars are shown for grid cells with 20% and 80% grasp success probability. Right: Center plot in 3D.

The next step is to transform the number of classification boundaries that include a grid cell into a probability value. This is done by computing the percentage of classification boundaries which surround a grid cell. In our example, the red grid cell has a grasp success probability of 0%, while the orange and green grid cells have grasp success probabilities of 35% and 100%. A grasp success probability of 35% for a grid cell means that the robot is able to grasp the object from 35% of the sampled object poses. For a high enough sample number, we consider that the sampled object poses are a good approximation of the real object pose. Therefore, we conclude that the robot will be able to successfully grasp the target object 35% of the time, when moving to the grid cell and performing the grasping action from there. The center image of Figure 4.9 depicts grasp success probability. Grid cells with low grasp success probability are colored red. White and green grid cells have successively higher grasp success probability. Please note, that the orange grid cell from the left image falls into the white region. We used orange color in the left image only for visual distinction from its surrounding white grid cells. The right image of Figure 4.9 depicts grasp success probability from an isometric perspective.

Please note the steep decline on the right side of the probability distribution near the table. Grasp success probability drops from 80% to 20% in just 5cm. This is intuitive because the table is located on the right side, and in simulation the robot bumped into the table always when moving too far to the right, leading to an unsuccessful experiment. Therefore, none of the 20 boundaries contain this area, and the variation in $\mathbf{P}$ on the right side of the PDM is low. Variations in $\mathbf{B}$ do not have a large effect on this boundary, as can be seen in the left image of Figure 4.9, where the classification boundaries mostly overlap on the right side, while they are spread out more on the other sides. However, when the robot stays just a small distance away from the table it does not bump into it and is able to successfully perform the grasp. When summing over the sampled boundaries, this leads to a steep decline in the probability distribution.

4.2.2.3 Uncertainty in Self-Localization

The robot is not only uncertaint about the target object's pose, but also about its own base position. This uncertainty is specified by $\langle \sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}} \rangle$ and has to be taken into account. For example, although all grid cells near to the left of the steep decline in the center image of Figure 4.9 have high grasp success probability, performing manipulation actions from there might still fail if there is a localization error and the robot's real base position is actually more to the right than expected. This would make the robot bump into the table unexpectedly. Therefore, the robot considers the uncertainty into its base position and analyzes how its real base position might be distributed around the estimated mean position. This is done by selecting 100 random samples in the x-y-plane and assigning the samples to the grid cell that contain them. For example, when the grid size is 2.5x2.5cm, then a random sample $\langle 0.04\text{m}, -0.01\text{m} \rangle$ would be assigned to the grid cell where the following condition holds

$$\Delta x_{\text{rob}} \in [0.025\text{m}, 0.05\text{m}[\;\wedge\; \Delta y_{\text{rob}} \in [-0.025\text{m}, 0.0\text{m}[$$

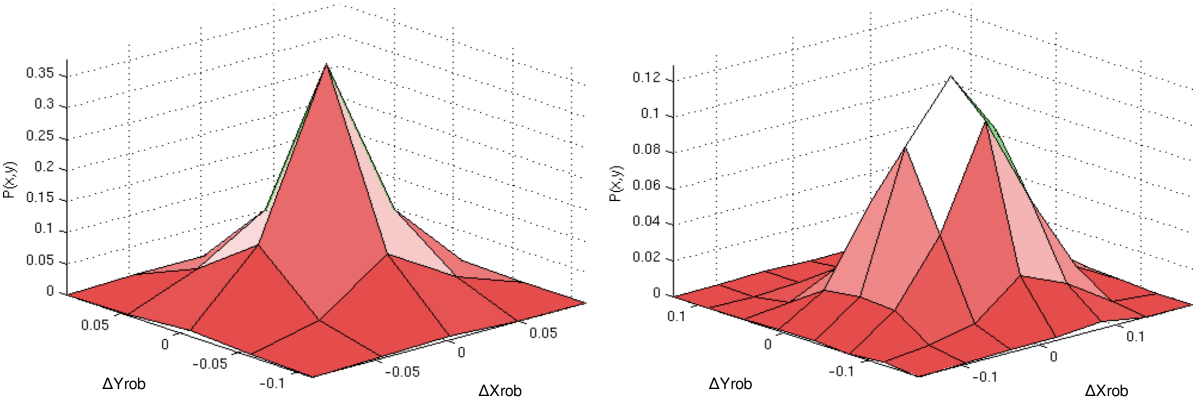


FIGURE 4.10 Probability distributions that were sampled from the robot's uncertainty into its base position. The probability distributions are centered around the robot's estimated mean base position. Left: $\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}} = 0.03$m. Right: $\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}} = 0.05$m.

Two examples of such probability distributions can be seen in Figure 4.10. The probability value at coordinate $\langle 0.0\text{m}, 0.0\text{m}\rangle$ represents the probability that the estimated mean base position is the robot's true base position. When the robot's uncertainty is low, then the probability distribution is more peaked around the estimated mean position. This can be seen in Figure 4.10, where the maximal probability value for an uncertainty of 0.03m nearly 35% (left image), while the probability distribution in the right image reaches only 12% (right image). Additionally, high uncertainties lead to probability plots with more grid cells, meaning that it is possible that the robot's real base position is farther off. In the left image of Figure 4.10 the probability distribution has a grid size of 4×4 grid cells, while the right image has a grid size of 6×6.

The probability distribution of the robot's base position is then conditioned with the grasp success probability distribution. The conditioning process is visualized in Figure 4.11. In this example, the robot's uncertainty into the target object's pose was $\sigma_{\Delta x_{\text{obj}}} = \sigma_{\Delta y_{\text{obj}}} = 0.02\text{m}$ and the uncertainty into its own pose was $\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}} = 0.03\text{m}$. It can be seen that the conditioning leads to smoother borders of the probability distribution. The conditioning step also works for multi-modal distributions as returned by particle filters.

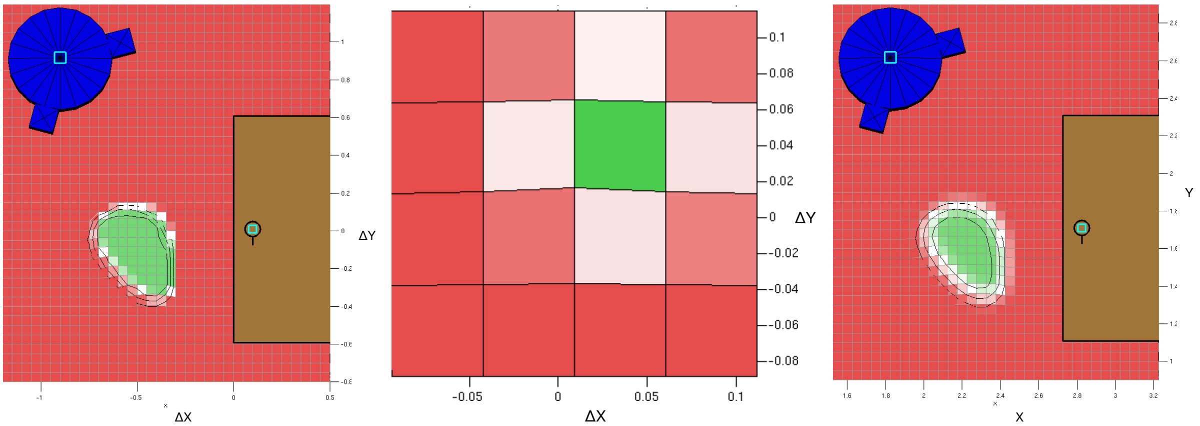


FIGURE 4.11 Conditioning of an ARPLACE probability distribution with a probability distribution of the robot's base position. Cyan rectangles depict the robot's uncertainty into the target object's pose and into its own pose. The bigger the rectangle, the bigger the uncertainty. Left: Grasp success probability for grasping the cup. Center: Probability distribution of robot's base positions. Right: Resulting probability distribution after convolving the prior distributions. Coordinates were transformed from the relative feature space to the world frame.

The probability distribution that results after the conditioning step is an ARPLACE probability distribution. It is specified with respect to the relative feature space. To be useful for the robot's navigation system, the ARPLACE is transformed so that it is relative to the world frame. This coordinate transformation was already performed in the right image of Figure 4.11.

An overview of the whole process of computing ARPLACE probability distributions is presented in algorithm 2.

```
input  : gsm = $\langle \overline{\mathbf{H}}, \mathbf{P}, \mathbf{W} \rangle$ ;            (generalized success model)
         robotposition uncertainty ;                  (covariance matrix)
         objectpose + uncertainty ;                   (mean pose, covariance matrix)
output : arplace ;                                    (probability distribution)
for i=1 to #samples do
    $\mathbf{t}_s$ = samplefromdistribution(objectpose,objectuncertainty);
    $\mathbf{b}_s = ([1\ \mathbf{t}_s] \cdot \mathbf{W})^T$;
    classif_boundary_set.add( $\overline{\mathbf{H}} + \mathbf{P} \cdot \mathbf{b}_s$ );
end
arplace = $\sum_{i=1}^{\#samples}$ grid(boundary_set$_i$) / #samples;
arplace = arplace * robotposition uncertainty ;       (Conditioning)
```

Algorithm 2: Pseudo-code for computing ARPLACE probability distributions.

4.2.2.4 Performance Analysis

The online computation of ARPLACEs is fast. Figure 4.12 depicts the time that is required for computing the ARPLACE from Figure 4.11. The ARPLACE was computed for grid cell sizes of 5cm x 5cm and 2.5cm x 2.5cm. The first number represents the runtime that is averaged over 100 experiments, and the value in brackets is the standard deviation. It can be seen that the computation of an ARPLACE takes approximately 44 milliseconds for a grid cell size of 5cm x 5cm which allows to compute ARPLACEs with 23 Hz. Querying the Generalized Success Model in order to recunstruct classification boundaries, which is done in step 1., is extremely fast with only 2 milliseconds. This is one of the main reasons why we learned a Generalized Success Model. Runtimes of step 3. and step 4. are even lower. Overall runtime of the algorithm is dominated by step 2. where it is evaluated in how many classification boundaries every grid cell is included. This requires looping over all grid cells and all classification boundaries. Looping however, is relatively slow in Matlab and it would be interesting to see what speedup can be achieved by implementing this piece of code in C. Standard deviations are low for all steps. To sum up, the computation of an Action-Related Place is fast. Fast enough for supporting least commitment planning. This is one of the main reasons why we chose the presented approach.

Computational Step	**Time** (in ms)	**Time** (in ms)
	Grid size 5 x 5cm	Grid size 2.5 x 2.5cm
1. Reconstruct classification boundaries	2.1 (0.2)	2.2 (0.4)
2. Sum over classification boundaries	41.1 (1.6)	114.2 (3.3)
3. Robot localization uncertainty	0.6 (0.5)	1.0 (0.1)
4. Conditioning	0.0 (0.0)	0.0 (0.0)
Overall	**44.0 (1.6)**	**117.3 (3.3)**

Figure 4.12 Mean of runtime for computing 100 ARPlaces. Runtimes are depicted for a grid cell size of 5cm x 5cm (2nd column) and 2.5cm x 2.5cm (third column). Numbers in brackets are standard deviations.

4.3 Evaluation

In this section, we evaluate the impact of all task-relevant parameters on an ARPlace probability distribution. Please remember that there are the following task-relevant parameters, namely A) object pose (variables (1) - (3) below); B) uncertainty into object pose (variables (4) and (5) below); C) uncertainty into robot position (variables (6) and (7) below).

- (1) Δx_{obj} : object's pose along x-axis of relative feature space (distance to table edge)
- (2) $y_{\text{obj}}^{\text{tab}}$: object's pose along table edge with respect to table frame
- (3) $\Delta\psi_{\text{obj}}$: object's angle with respect to relative feature space
- (4) $\sigma_{\Delta x_{\text{obj}}}$: uncertainty into object's distance to table edge
- (5) $\sigma_{\Delta y_{\text{obj}}}$: uncertainty into object's pose along table edge
- (6) $\sigma_{\Delta x_{\text{rob}}}$: uncertainty into robot's base position along x-axis
- (7) $\sigma_{\Delta y_{\text{rob}}}$: uncertainty into robot's base position along y-axis

We perform seven experiments, and in each experiment one task-relevant parameter is varied while the other six parameters are fix. The resulting ARPlace probability distributions will illustrate the effect of the varied parameter. For clarity, we exclusively consider ARPlace probability distributions for grasping from the side and for grasping from table edge 1.

4.3.1 Impact of Object's Distance to Table Edge

In this section we vary the distance of the cup to the table edge Δx_{obj}. The other parameters are set to the following default values: (2) $y_{\text{obj}}^{\text{tab}} = 0.00\text{m}$ which means that the cup's distance

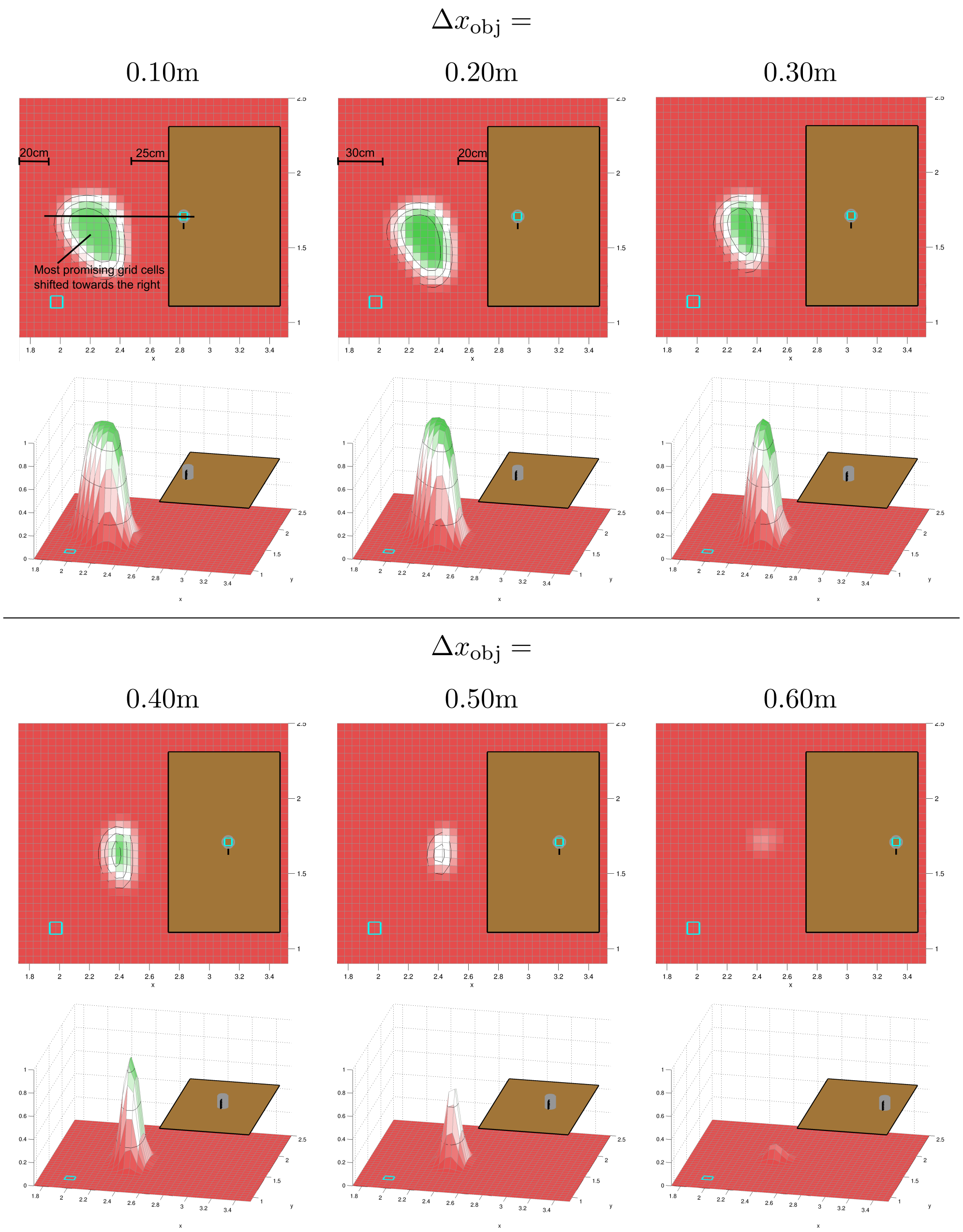


FIGURE 4.13 Visualization of how the ARPLACE changes for different values of Δx_{obj}. Every ARPLACE is visualized from a top-down (first and third image row) as well as an isometric perspective (second and fourth image row). The value of Δx_{obj} for two corresponding ARPLACE plots is denoted above the plots. Uncertainty into the object's pose is depicted by the cyan square in the cup's center. Uncertainty into the robot's base position is depicted by the cyan square in the left bottom corner. Further annotations are explained in the text.

to table edge 2 and table edge 4 is equal; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$ which means that the cup's handle is pointing down, parallel to table edge 1; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$ indicating small uncertainty of the robot into Δx_{obj}; (5) $\sigma_{y^{\mathrm{tab}}_{\mathrm{obj}}} = 0.025\mathrm{m}$ indicating small uncertainty of the robot into Δy_{obj}; (6) $\sigma_{\Delta x_{\mathrm{rob}}} = 0.04\mathrm{m}$ indicating medium uncertainty of the robot into its base position along the x-axis of the relative feature space; (7) $\sigma_{\Delta y_{\mathrm{rob}}} = 0.04\mathrm{m}$ indicating medium uncertainty of the robot into its base position along the y-axis of the relative feature space. Figure 4.13 depicts ARPLACE probability distributions for the following values of Δx_{obj}

$$\Delta x_{\mathrm{obj}} \in \{0.10\mathrm{m}, 0.20\mathrm{m}, 0.30\mathrm{m}, 0.40\mathrm{m}, 0.50\mathrm{m}, 0.60\mathrm{m}\}$$

For developing a better intuition of how ARPLACE probability distributions look a top-down view and an isometric perspective is presented for every value of Δx_{obj}. Uncertainties are drawn to scale, so a standard deviation of 0.025m leads to a square where each edge is 0.05m long. That is why the cup's uncertainty can be seen only in the top-down view. Because the radius of the cup is 4cm, the square representing cup pose uncertainty is hidden by the cup's body, when viewed from an isometric perspective. The black isobars within ARPLACE probability distributions refer to grasp success probability levels of 20%, 50%, and 80%.

Several observations can be made when analyzing Figure 4.13. When looking at the ARPLACEs where $x_{\mathrm{obj}} = 0.10\mathrm{m}$, then several grid cells with a grasp success probability of more than 80% can be seen. Therefore, the corresponding task seems to be not very challenging. The reason is that the target object is well in reach, because it is near the table edge. Moreover, the uncertainties into the object's pose and into the robot's base position are relatively low. Good state estimations simplify the search for good manipulation places, and as a result the manipulation action is more likely to succeed.

When the object's distance to the table edge increases, grasping becomes more difficult. When comparing the images where $\Delta x_{\mathrm{obj}} = 0.10\mathrm{m}$ and $\Delta x_{\mathrm{obj}} = 0.20\mathrm{m}$, it can be seen that the cup's distance to the table edge got bigger. The ARPLACE probability distribution for $\Delta x_{\mathrm{obj}} = 0.20\mathrm{m}$ shifted towards the table edge while it has almost the same shape as for $\Delta x_{\mathrm{obj}} = 0.10\mathrm{m}$. The reason is that the ARPLACE tried to maintain its relative position with respect to the object. Therefore, the distance of the ARPLACE to the left image border shifted by exactly the same distance that the cup shifted: 10cm. The distance of the ARPLACE to the table edge only decreased from 25cm to 20cm, as is illustrated in the center top image of Figure 4.13. The Generalized Success Model does not allow the robot to move closer to the table, because it learned that the robot may bump into it which leads to manipulation failure. That is why the ARPLACE got a bit compressed on the right side, although this is hardly visible in the image for $\Delta x_{\mathrm{obj}} = 0.20\mathrm{m}$.

When the object's distance to the table edge increases even more, then the compression becomes more obvious. When $\Delta x_{\mathrm{obj}} = 0.30\mathrm{m}$, for example, this requires the robot to perform the manipulation action even closer to the table. The distance of the ARPLACE probability distribution to the table edge however, remained at 20cm which is the case for all higher values of Δx_{obj} as well. For $\Delta x_{\text{obj}} = 0.40\text{m}$ the area of promising manipulation places gets significantly smaller. At least there are some remaining grid cells with a grasp success probability of more than 80%. When $\Delta x_{\text{obj}} = 0.50\text{m}$, then the grasping task gets considerably more difficult, reaching a maximum grasp success probability of just above 50%. When trying to grasp the cup for $\Delta x_{\text{obj}} = 0.60\text{m}$, then the robot rightfully expects that the manipulation action will most likely fail, even when performed from the most promising base position.

The question could arise, why the ARPLACE probability distribution in the above plots does maintain a minimum distance of just 20cm, while the robot's radius is 25cm. The answer is the uncertainty of the robot into its base position. The conditioning process stretches and smoothes the borders of an ARPLACE probability distribution into any direction.

Another observation is that the most promising grid cells are in front of the cup, or shifted a bit to the right. This can be seen when imagineing a line that is parallel to the x-axis of the relative feature space and goes through the center of the cup, as depicted in the top left image of Figure 4.13. Then the most promising grid cells are near to that line or slightly more to the right of it when looking at the cup from the ARPLACE probability distribution. The reason for that is the reaching trajectory that is generated by the vector-field controller as depicted in Figure 3.3. The arm first moves down and to the side of the cup's handle while making the gripper parallel to the handle. Then the gripper moves inwards in order to position the gripper around the handle. When using the right arm for grasping, then it is preferable to start this type of reaching motion with the arm being to the right of the target object. This gives the arm more space to smoothly glide inwards in order to position the gripper around the target object. The reason why the ARPLACE probability distribution is not shifted even more to the right is because ARPLACEs capture the probability that the manipulation action will succeed, when the robot performs the manipulation from a *base position* within the corresponding grid cell. The arms of the robot, however, are not mounted in the center of the robot's base but slightly to the front and to the side as can be seen in Figure A.2. This brings the arm another 12cm to the right of the cup.

The left plot of Figure 4.14 depicts maximum grasp success probability for different values of Δx_{obj}. It can be seen that grasp success probability is very high for values between 0.10m and 0.42m. When the object's distance to the table edge gets bigger than $\approx 0.42\text{m}$, then maximal grasp success probability shrinks significantly which is intuitive because the object is hardly reachable any more. It is not intuitive that values of 0.09m and below do not lead to higher grasp success probability. The reason is that our minimum value for Δx_{obj} when learning the Generalized Success Model was 0.07m. Values that are smaller than 0.07m are therefore outside the parameter space of the Generalized Success Model and therefore no classification boundaries were learned for them. When learning future versions of Generalized Success Models this should be taken into account and the minimum value for Δx_{obj} should be reduced.

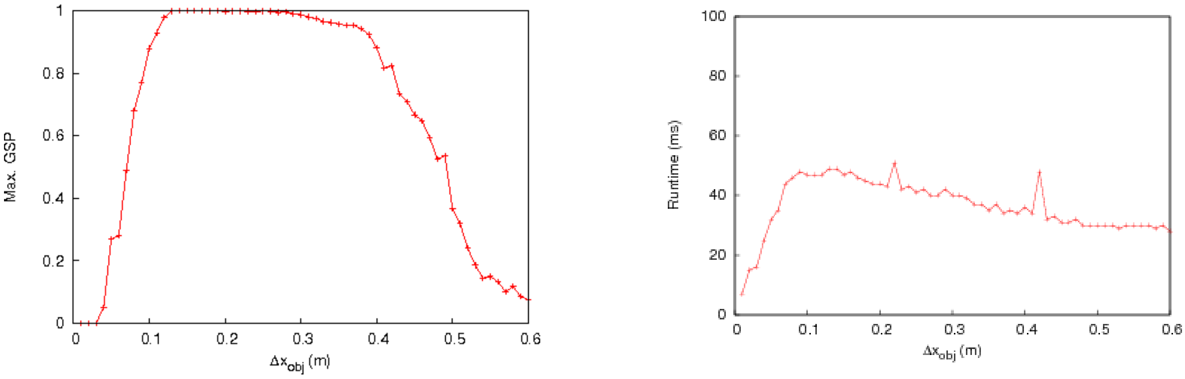


FIGURE 4.14 Maximum grasp success probability (left plot) and runtime (right plot) for $\Delta x_{\text{obj}} \in \{0.00\text{m}, .., 0.60\text{m}\}$.

The right plot of Figure 4.14 depicts the runtime for computing ARPlaces. It can be seen that the runtime peaks at 44ms for $x_{\text{obj}} \in \{0.09\text{m}, .., 0.24\text{m}\}$. The same values where grasp success probability reaches its maximum. For smaller values of x_{obj}, the the runtime is low. The reason is that small values of x_{obj} are outside the parameter space of the Generalized Success Model. As a consequence, no classification boundaries are reconstructed for these inputs. Fewer classification boundaries, however, lead to low runtime because the most time consuming step is to sum over classification boundaries.

4.3.2 Impact of Object's Distance along Table Edge

In this section we vary the distance of the cup along the table edge (task-relevant parameter (2) $y_{\text{obj}}^{\text{tab}}$. The other parameters are set to their default values: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (3) $\Delta \psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$.

Figure 4.15 depicts ARPlace probability distributions for the following values of y_{obj}

$$y_{\text{obj}}^{\text{tab}} \in \{-0.20\text{m}, 0.00\text{m}, 0.20\text{m}\}$$

It can be seen that the ARPlace distribution includes many base positions with high grasp success probability. Because the cup pose only moves along the table edge, the ARPlace probability distribution remains its shape, while moving along the table edge by the same

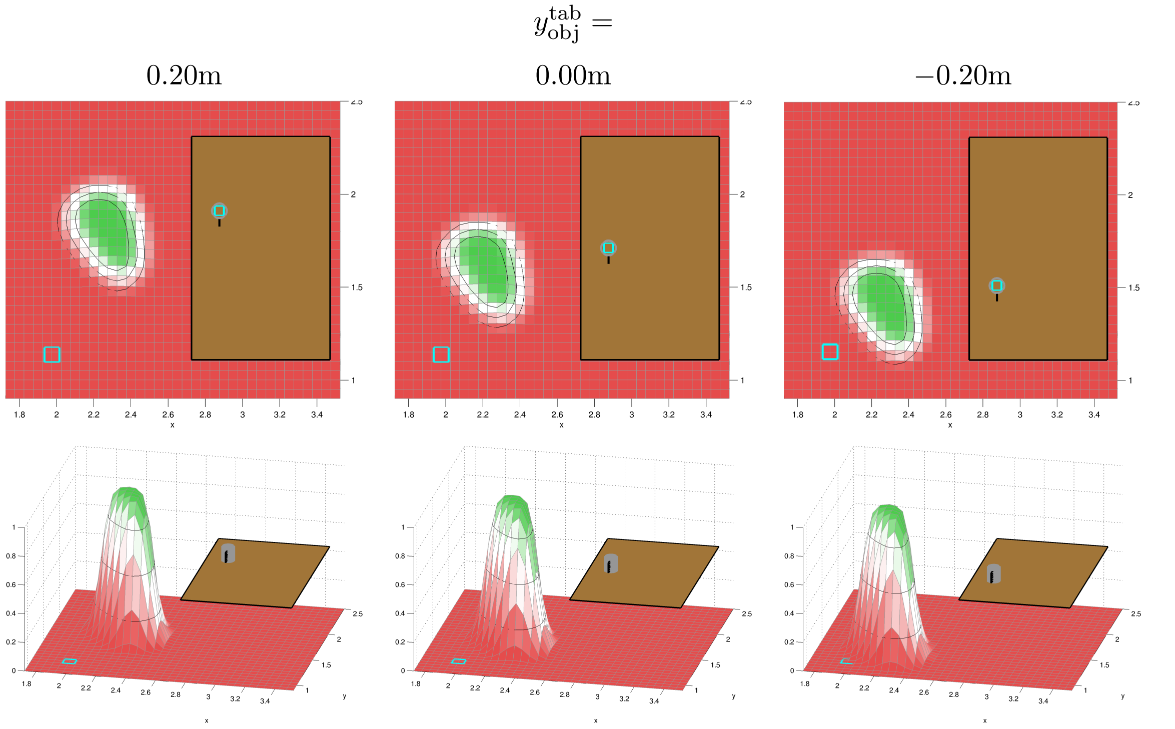


FIGURE 4.15 Visualization of how the ARPLACE changes for different values of $y_{\text{obj}}^{\text{tab}}$. Every ARPLACE is visualized from a top-down (first image row) and an isometric perspective (second image row).

amount as the object. More precisely: The cup's pose between the images for $y_{\text{obj}}^{\text{tab}} = 0.20\text{m}$ and $y_{\text{obj}}^{\text{tab}} = 0.00\text{m}$ changes by 20cm towards the bottom of the image. The corresponding ARPLACE probability distribution changes by exactly the same vector. Maximum grasp success probability is similar in every plot. This is as expected. Because $y_{\text{obj}}^{\text{tab}}$ only shifts classification boundaries *after* they have been reconstructed, $y_{\text{obj}}^{\text{tab}}$ does influence the position of the ARPLACE probability distribution, but not its shape. When ARPLACEs for different values of $y_{\text{obj}}^{\text{tab}}$ look a bit different, then this is due to variations in the sampling process.

Figure 4.16 depicts maximum grasp success probability and runtime for several ARPLACEs with different values of $y_{\text{obj}}^{\text{tab}}$. Maximum grasp success probability is between 98.9% and 100% and runtime is between 47ms and 51ms. Please note that the first experiment for $y_{\text{obj}}^{\text{tab}} = -0.20\text{m}$ had a runtime of 72ms. Because this outlier was peculiar we repeated the experiment with the same outcome. After that we found that runtime was longer because of memory allocation operations that were not necessary for subsequent experiments. That is why we perform the first experiment twice and store only the runtime of the second experiment.

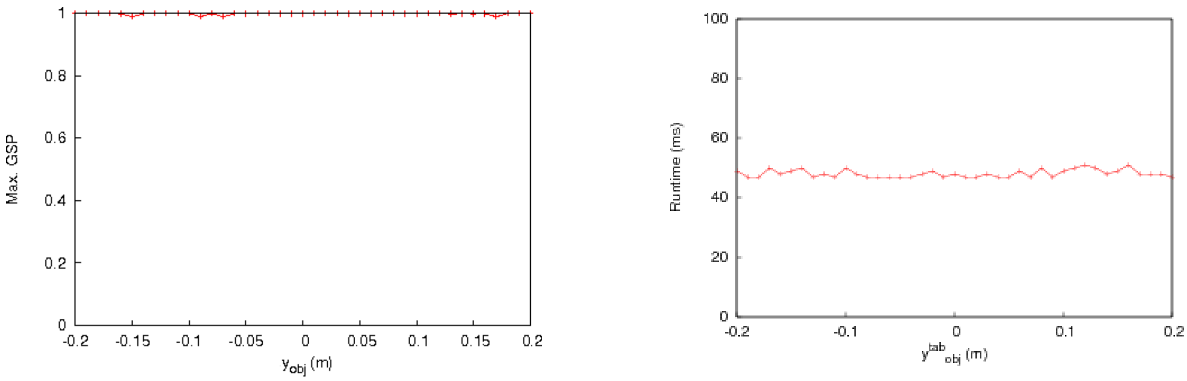


FIGURE 4.16 Maximum grasp success probability (left plot) and runtime (right plot) for $y^{\mathrm{tab}}_{\mathrm{obj}} \in \{-0.20\mathrm{m}, .., 0.20\mathrm{m}\}$.

4.3.3 Impact of Object Orientation

In this section we vary the orientation of the cup (task-relevant parameter (3) $\Delta\psi_{\mathrm{obj}}$). The other parameters are set to their default values: (1) $\Delta x_{\mathrm{obj}} = 0.15\mathrm{m}$; (2) $y^{\mathrm{tab}}_{\mathrm{obj}} = 0.00\mathrm{m}$; (4) $\sigma_{\Delta x_{\mathrm{obj}}} = 0.025\mathrm{m}$; (5) $\sigma_{\Delta y_{\mathrm{obj}}} = 0.025\mathrm{m}$; (6) $\sigma_{\Delta x_{\mathrm{rob}}} = 0.04\mathrm{m}$; (7) $\sigma_{\Delta y_{\mathrm{rob}}} = 0.04\mathrm{m}$.

Figure 4.17 depicts ARPLACE probability distributions for the following values of $\Delta\psi_{\mathrm{obj}}$

$$\Delta\psi_{\mathrm{obj}} \in \{-\frac{9}{10}\pi, -\frac{2}{3}\pi, -\frac{1}{3}\pi, 0, \frac{1}{3}\pi, \frac{2}{3}\pi\}$$

For $\Delta\psi_{\mathrm{obj}} = -\frac{9}{10}\pi$ the ARPLACE probability distribution represents grasp success probability for grasping the target object with the right arm. There is no grid cell with grasp success probability of above 0% for the left arm. The reason is that the Powercube arm is not dextrous enough. The ARPLACE for $\Delta\psi_{\mathrm{obj}} = -\frac{9}{10}\pi$ is different to the ARPLACEs we saw until now in that it is rounder, more far away from the table, and shifted more to the left of the cup (when the cup is seen from a position in front of the table). Until now we always saw ARPLACEs that were shifted towards the right of the cup in order to enable the arm to smoothly approach the cup's handle. But in previous scenarios the handle was always oriented so that it pointed to the right ($\Delta\psi_{\mathrm{obj}} = -\frac{\pi}{2}$). Now the handle is oriented so that it nearly points towards the robot. When using our vector field controller for grasping, this leads to a different approach vector. The vector field controller adapts the gripper's orientation so that it is parallel to the handle, and then approaches the handle parallel to the grasping point. The approach vector for $\Delta\psi_{\mathrm{obj}} = -\frac{9}{10}\pi$ is from the front of the cup instead of from the right side, as depicted in the top left image of Figure 4.17. Therefore, the robot needs more space towards the front and less space to the right side. That is why the most promising base positions are found more far away from the table, and more to the left of the cup.

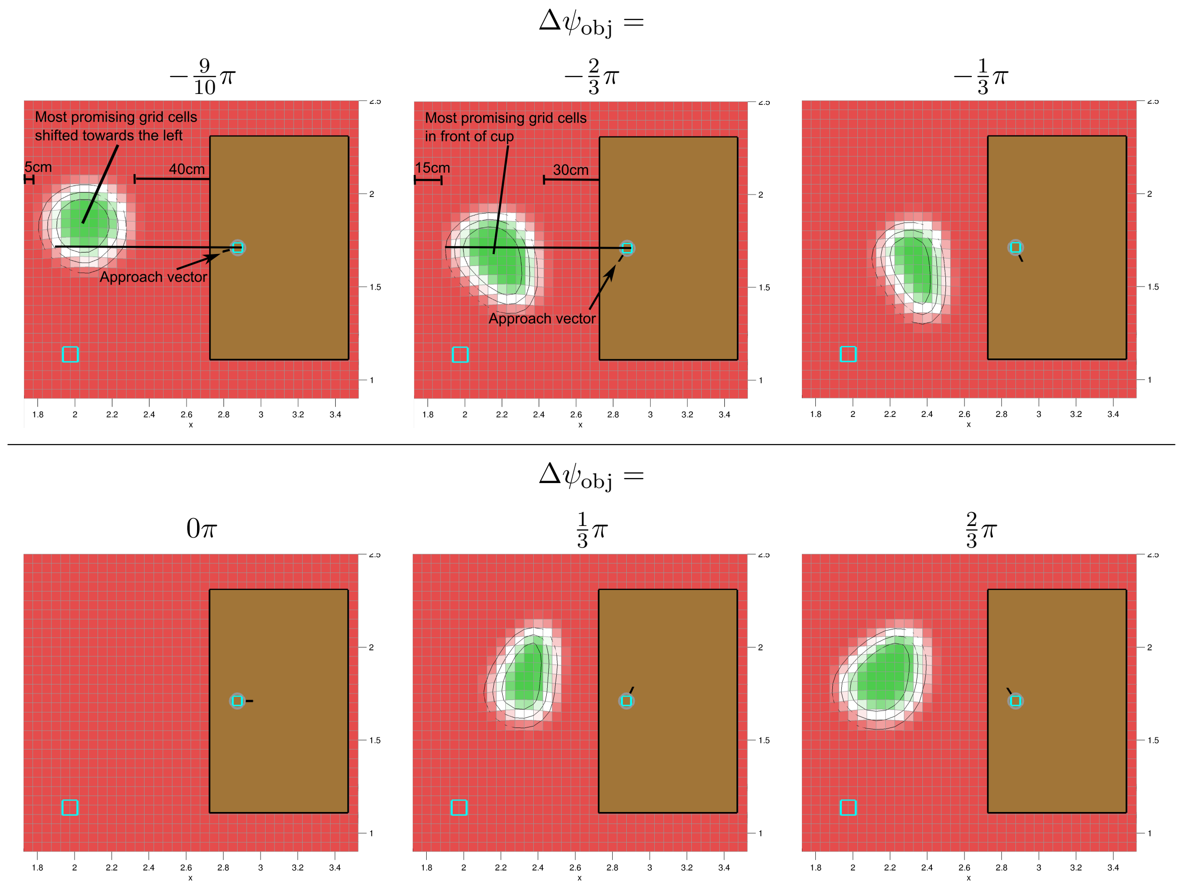


FIGURE 4.17 Visualization of how the ARPLACE changes for different values of $\Delta\psi_{\text{obj}}$. The approach vectors depict from which direction the robot approaches the handle with the gripper.

For $\Delta\psi_{\text{obj}} = -\frac{2}{3}\pi$, the approach vector of the gripper is shifted a bit towards the right. As a result, the ARPLACE probability distribution is closer to the table and shifts more to the right. The most promising base positions are now directly in front of the cup. This effect is amplified when the cup is rotated further to $\Delta\psi_{\text{obj}} = -\frac{1}{3}\pi$. Now the robot has to approach the cup from the back and from the side, requiring it to be even closer to the table because it has to reach behind the cup first in oder to approach it from the back. When $\Delta\psi_{\text{obj}} = 0\pi$ then the robot is not able to grasp the cup. Its manipulator is not dextrous enough to perform a satisfying reaching motion in this case.

Until now all ARPLACEs represented grasp success probability for grasping with the right arm. The ARPLACEs for the cup orientations $\Delta\psi_{\text{obj}} = \frac{1}{3}\pi$ and $\Delta\psi_{\text{obj}} = \frac{2}{3}\pi$ are for performing the grasp with the left arm. There is no grid cell with a grasp success probability of above 0% for the right arm. In fact, the ARPLACEs for $\Delta\psi_{\text{obj}} = \frac{1}{3}\pi$ and $\Delta\psi_{\text{obj}} = \frac{2}{3}\pi$ are mirrored versions of ARPLACEs for $\Delta\psi_{\text{obj}} = -\frac{1}{3}\pi$ and $\Delta\psi_{\text{obj}} = -\frac{2}{3}\pi$, but for grasping with the left instead of the right arm. This is intuitive, because the kinematics (axes of rotation, link lengths, joint limits) of the Powercube arms are mirrored versions of each other.

Figure 4.18 depicts maximum grasp success probability and runtime for several ARPLACEs with different values of $\Delta\psi_{\text{obj}}$. Because the robot is not able to grasp the cup when its handle points away from the robot ($\Delta\psi_{\text{obj}} \in [-0.503\text{rad}, .., 0.503\text{rad}]$), or when it directly faces the robot ($\Delta\psi_{\text{obj}} \in [-\pi, .., -3.016\text{rad}] \cup [3.016\text{rad}, .., \pi]$), no classification boundaries can be reconstructed in this case and grasp success probability is 0%. When no classification boundary can be reconstructed, grasp success probability is 0% everywhere and further computations are aborted.

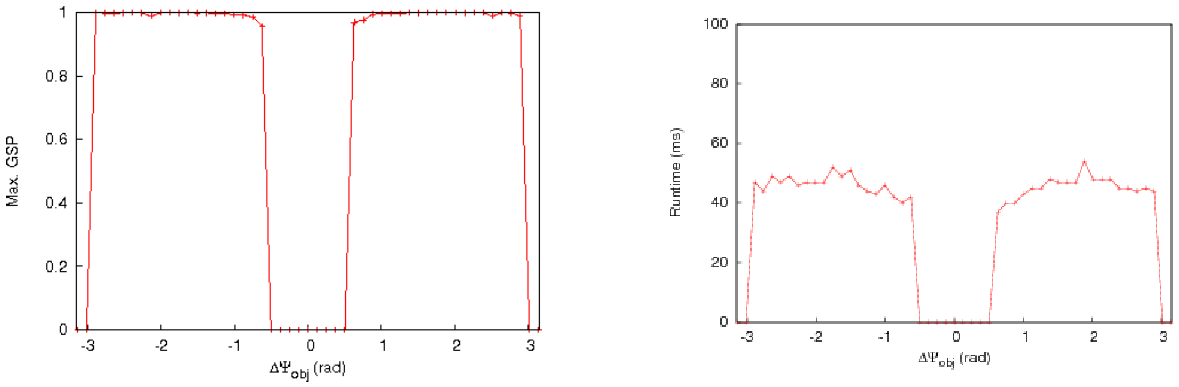


FIGURE 4.18 Maximal grasp success probability (left plot) and runtime (right plot) for $\Delta\psi_{\text{obj}} \in \{-\pi, .., \pi\}$.

4.3.4 Impact of Uncertainty into Object's Pose

In this section we vary the uncertainty of the robot into the object's pose, which is specified by the task-relevant parameters (4) $\sigma_{\Delta x_{\text{obj}}}$ and (5) $\sigma_{\Delta y_{\text{obj}}}$. First we evaluate the impact of $\sigma_{\Delta x_{\text{obj}}}$. The other parameters are set to their default values: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$.

Figure 4.19 depicts ARPLACE probability distributions for the following values of $\sigma_{\Delta x_{\text{obj}}}$

$$\sigma_{\Delta x_{\text{obj}}} \in \{0.00\text{m}, 0.04\text{m}, 0.08\text{m}, 0.16\text{m}, 0.23\text{m}, 0.30\text{m}\}$$

For $\sigma_{\Delta x_{\text{obj}}} = 0.00\text{m}$, the corresponding ARPLACE probability distribution has a large number of grid cells where grasp success probability is above 80%. It even reaches 100% at 3-4 grid cells, which is indicated by a plateau at the top.

For $\sigma_{\Delta x_{\text{obj}}} = 0.04\text{m}$, the ARPLACE is very similar, although the area where grasp success probability is above 80% is slightly smaller. This decrease in grasp success probability contin-

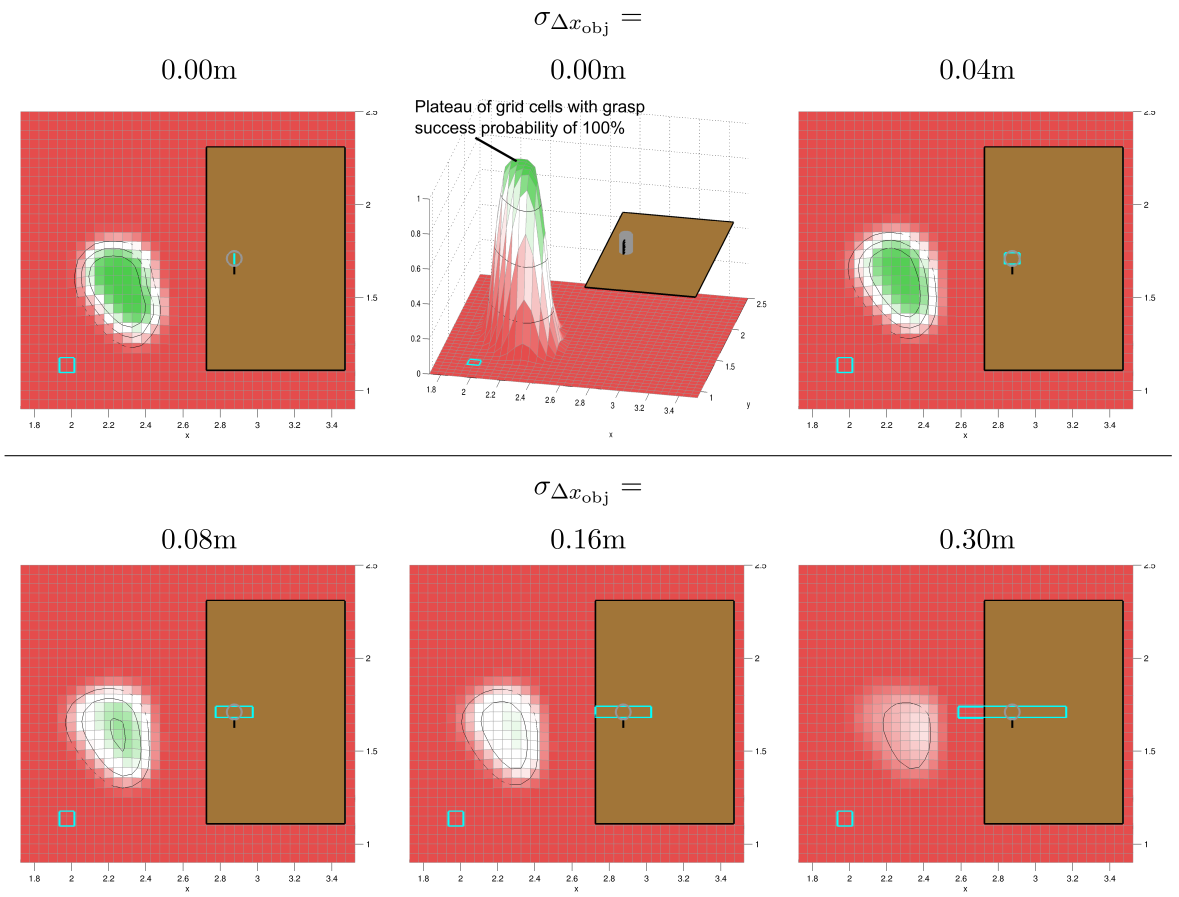


FIGURE 4.19 Visualization of how the ARPLACE changes for different values of $\sigma_{\Delta x_{\text{obj}}}$.

ues as $\sigma_{\Delta x_{\text{obj}}}$ increases. When $\sigma_{\Delta x_{\text{obj}}} = 0.08\text{m}$ there are only a few grid cells left with a grasp success probability of above 80%. For $\sigma_{\Delta x_{\text{obj}}} = 0.16\text{m}$ maximum grasp success probability reaches only 60%, and for a huge uncertainty of $\sigma_{\Delta x_{\text{obj}}} = 0.30\text{m}$ the most promising grid cells reach only 30% grasp success probability.

As expected, grasp success probability decreases when the uncertainty about the object's pose increases. The reason is that when the robot is uncertain about Δx_{obj}, it is hard to decide how close to move to the table. In Figure 4.19, the real pose of the target object could be anywhere within the cyan rectangle with a probability of 68.3% because the cyan rectangle visualizes the standard deviation of the object's pose. Even worse it could be outside the cyan rectangle with a probability of 31.7%. When the robot moves too close to the table and the target object is closer to the table edge than expected (the estimated value of Δx_{obj} is bigger than the true value of Δx_{obj}), then chances are high that the robot hits the cup with its gripper because there is too little space to execute the reaching trajectory that makes the gripper parallel to the handle. On the other side when the robot stays far away from the table, but the target object is more far away from the table edge than expected (the estimated value of Δx_{obj} is smaller than true value of Δx_{obj}), then the object might be out of reach.

Figure 4.20 depicts maximum grasp success probability and runtime for several ARPLACEs with different values of $\sigma_{\Delta x_{\text{obj}}}$. It can be seen that maximum grasp success probability decreases when $\sigma_{\Delta x_{\text{obj}}}$ increases. The correlation seems to be linear although there are some "jumps" which are most likely due to sampling variations. The level of maximum grasp success probability falls below 80% when $\sigma_{\Delta x_{\text{obj}}}$ exceeds 0.10m.

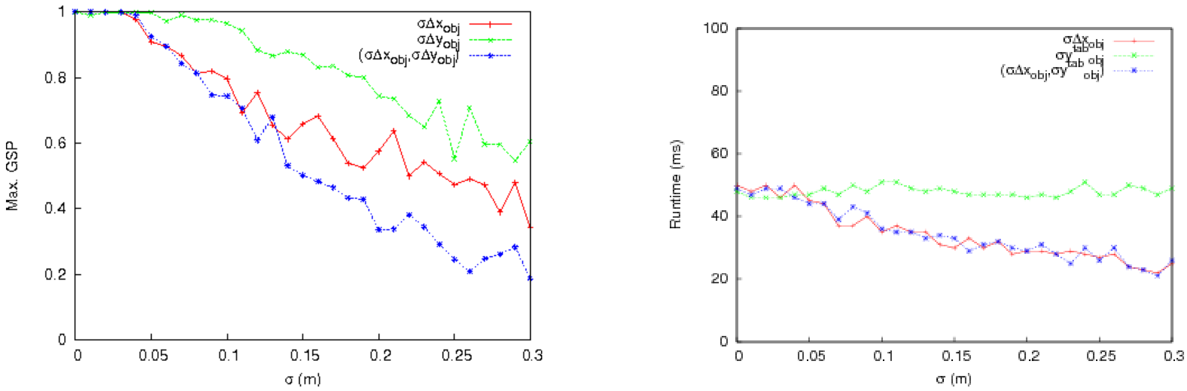


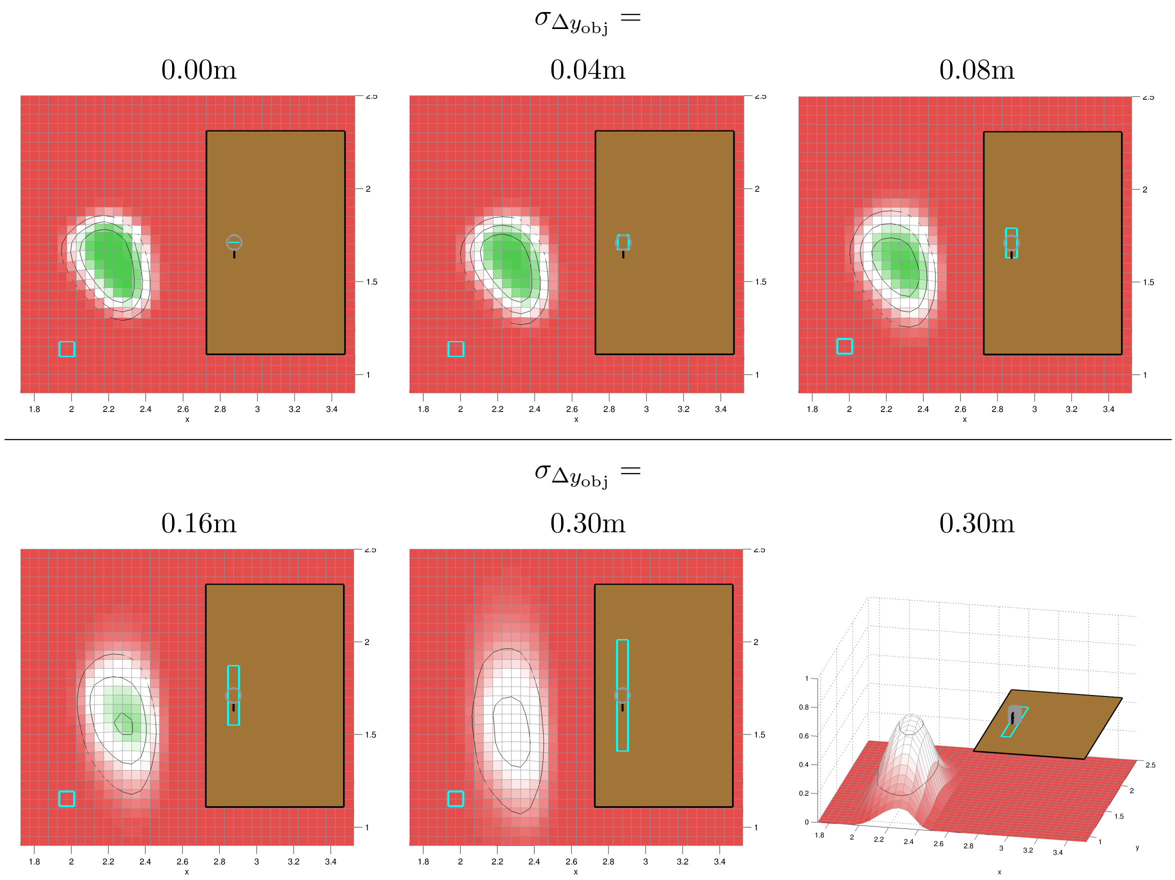
FIGURE 4.20 Maximal grasp success probability (left plot) and runtime (right plot) for $\sigma_{\Delta x_{\text{obj}}} \in \{0.00\text{m}, .., 0.30\text{m}\}$, $\sigma_{\Delta y_{\text{obj}}} \in \{0.00\text{m}, .., 0.30\text{m}\}$, and $(\sigma_{\Delta x_{\text{obj}}}, \sigma_{\Delta y_{\text{obj}}}) \in \{(0.00\text{m}, 0.00\text{m}), .., (0.30\text{m}, 0.30\text{m})\}$.

We continue with evaluating the impact of $\sigma_{\Delta y_{\text{obj}}}$ on the ARPLACE probability distribution. The other parameters are set to their default values: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$.

Figure 4.21 depicts ARPLACE probability distributions for the following values of $\sigma_{\Delta y_{\text{obj}}}$

$$\sigma_{\Delta y_{\text{obj}}} \in \{0.00\text{m}, 0.04\text{m}, 0.08\text{m}, 0.16\text{m}, 0.30\text{m}\}$$

For $\sigma_{\Delta y_{\text{obj}}} = 0.00\text{m}$, where the robot is completely certain about the object's pose along the table edge, the ARPLACE probability distribution has a large number of grid cells where grasp success probability is above 80%. The same is the case for $\sigma_{\Delta y_{\text{obj}}} = 0.04\text{m}$. For $\sigma_{\Delta y_{\text{obj}}} = 0.08\text{m}$ the most promising grid cells are still very good, although it is clearly visible that the number of grid cells with maximal grasp success probability of more than 80% decreases. While maximum grasp success probability decreases, the ARPLACE probability distribution gets bigger, meaning that there are more grid cells with a grasp success probability above 0%. This is most noticeable at the borders of the ARPLACE that stretch out more than previously. The decrease of maximum grasp success probability and increase of ARPLACE area continues for $\sigma_{\Delta y_{\text{obj}}} = 0.16\text{m}$ and $\sigma_{\Delta y_{\text{obj}}} = 0.30\text{m}$.



$\sigma_{\Delta y_{\mathrm{obj}}} =$

0.00m 0.04m 0.08m

$\sigma_{\Delta y_{\mathrm{obj}}} =$

0.16m 0.30m 0.30m

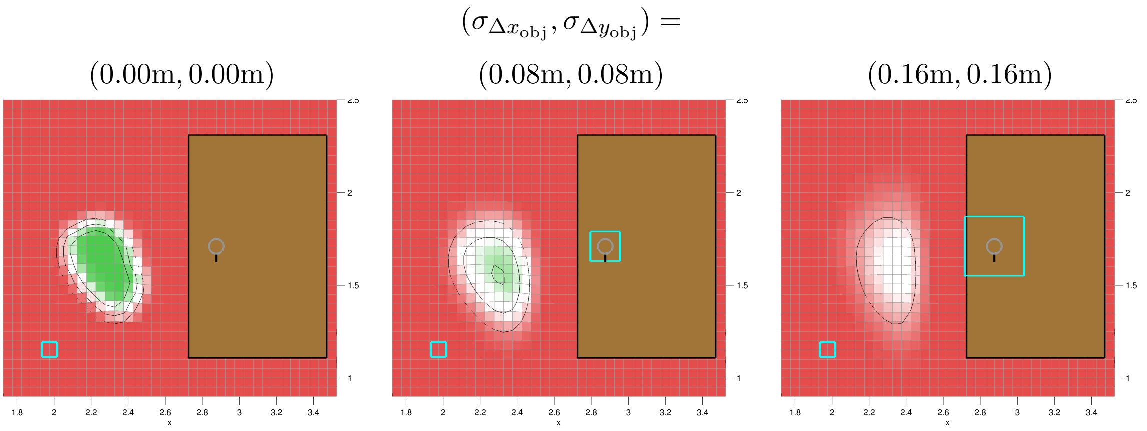
FIGURE 4.21 Visualization of how the ARPLACE changes for different values of $\sigma_{\Delta y_{\mathrm{obj}}}$.

Figure 4.20 depicts maximum grasp success probability for several values of $\sigma_{\Delta y_{\mathrm{obj}}}$. The correlation between maximum grasp success probability and $\sigma_{\Delta y_{\mathrm{obj}}}$ seems to be linear and there are fewer “jumps” than for the plot of $\sigma_{\Delta x_{\mathrm{obj}}}$. The level of grasp success probability falls below 80% when $\sigma_{\Delta y_{\mathrm{obj}}}$ exceeds 0.19m. The plot further reveals that maximum grasp success probability is always higher when the uncertainty of $\sigma_{\Delta y_{\mathrm{obj}}}$ and $\sigma_{\Delta x_{\mathrm{obj}}}$ is set to the same value. This result suggests that the robot is more robust towards uncertainties of $\sigma_{\Delta y_{\mathrm{obj}}}$ than it is towards uncertainties of $\sigma_{\Delta x_{\mathrm{obj}}}$. Or less formal: the robot can handle uncertainties of the object’s pose along the table edge better than it can handle uncertainties of the object’s distance to the table edge. Another hint is given when comparing Figure 4.21 to Figure 4.19. For example, maximum grasp success probability for $\sigma_{\Delta y_{\mathrm{obj}}} = 0.16\mathrm{m}$ is above 80%, while maximum grasp success probability for $\sigma_{\Delta x_{\mathrm{obj}}} = 0.16\mathrm{m}$ is below 80%. Actually, the ARPLACE probability distribution for $\sigma_{\Delta y_{\mathrm{obj}}} = 0.16\mathrm{m}$ is very similar to that of $\sigma_{\Delta x_{\mathrm{obj}}} = 0.08\mathrm{m}$ when comparing maximum grasp success probability.

The reason may be that the robot’s motion system can compensate for errors along table edges more easily by stretching the arm out farther to the left or right. When considering the robot's arm length of 0.84cm to the front of the gripper, then the robot can stretch out his arm 0.84cm to either direction. So the span width of the arm along the table edge is 1.68m. The span width of the arm from the front to the back is also 1.68cm. However, for the task of grasping the cup, only the span width to the front is relevant, which is only 0.84m. Although we have not proven this yet, our claim is that the limited span width is the reason why the robot can handle uncertainties of $\sigma_{\Delta y_{\mathrm{obj}}}$ more robustly than uncertainties of $\sigma_{\Delta x_{\mathrm{obj}}}$.

To conclude this section, we evaluate the impact of varying $\sigma_{\Delta x_{\mathrm{obj}}}$ and $\sigma_{\Delta y_{\mathrm{obj}}}$ simultaneously. Figure 4.22 depicts ARPLACE probability distributions for the following values of $(\sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}})$

$$(\sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}}) \in \{(0.00\mathrm{m}, 0.00\mathrm{m}), (0.08\mathrm{m}, 0.08\mathrm{m}), (0.16\mathrm{m}, 0.16\mathrm{m})\}$$



$(\sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}}) =$

$(0.00\mathrm{m}, 0.00\mathrm{m})$ $(0.08\mathrm{m}, 0.08\mathrm{m})$ $(0.16\mathrm{m}, 0.16\mathrm{m})$

FIGURE 4.22 Visualization of how the ARPLACE changes for different values of $(\sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}}$.

Maximum grasp success probability for more values of $(\sigma_{\Delta x_{\mathrm{obj}}}, \sigma_{\Delta y_{\mathrm{obj}}})$ and the runtime for computing them is depicted in Figure 4.20.

4.3.5 Impact of Uncertainty into Robot's Pose

In this section we vary the uncertainty of the robot into its base position, which is specified by the task-relevant parameters (6) $\sigma_{\Delta x_{\mathrm{rob}}}$ and (7) $\sigma_{\Delta y_{\mathrm{rob}}}$. We start with evaluating the impact of $\sigma_{\Delta x_{\mathrm{rob}}}$. The other parameters are set to their default values: (1) $\Delta x_{\mathrm{obj}} = 0.15\mathrm{m}$; (2) $y^{\mathrm{tab}}_{\mathrm{obj}} = 0.00\mathrm{m}$; (3) $\Delta\psi_{\mathrm{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\mathrm{obj}}} = 0.025\mathrm{m}$; (5) $\sigma_{\Delta y_{\mathrm{obj}}} = 0.025\mathrm{m}$; (7) $\sigma_{\Delta y_{\mathrm{rob}}} = 0.04\mathrm{m}$.

Figure 4.23 depicts ARPLACE probability distributions for the following values of $\sigma_{\Delta x_{\mathrm{rob}}}$

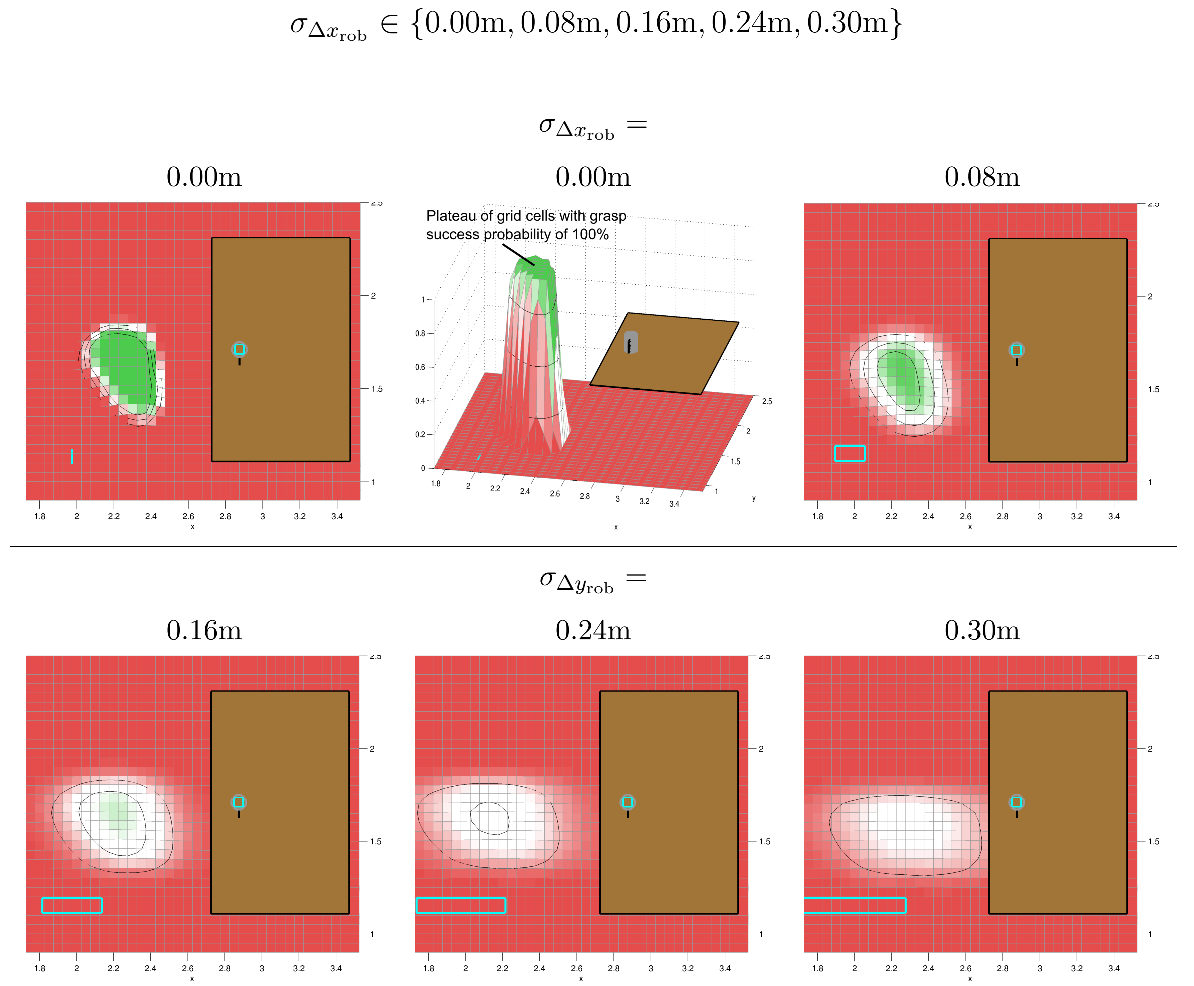


FIGURE 4.23 Visualization of how the ARPLACE changes for different values of $\sigma_{\Delta x_{\text{rob}}}$.

For $\sigma_{\Delta x_{\text{rob}}} = 0.00\text{m}$, the corresponding ARPLACE probability distribution has a large number of grid cells where grasp success probability is above 80%. It even reaches 100% at several grid cells, which is indicated by a plateau at the top. The plateau is more noticeable than the plateau in Figure 4.19. An answer to the question why the plateau is more pronounced is not trivial. An easy but incorrect answer to the question of the more pronounced plateau would be that the default object pose uncertainty (0.025m) in the current case is lower than the default base position uncertainty in the case of Figure 4.19 (0.04m), resulting in an ARPLACE probability distribution with higher maximum grasp success probability. However, the impact of object pose uncertainty and base position uncertainty can not be compared. Object pose uncertainty is taken into account when reconstructing classification boundaries. Figure 4.8 depicts that even if there is a considerable amount of object pose uncertainty, there are still several grid cells that are included in *all* classification boundaries, leading to a grasp success probability of 100%. Only if object pose uncertainty raises above a certain threshold, then there is no grid cell anymore that is included in all reconstructed classification boundaries. On the other side, when base position uncertainty increases, then it *immediately* impacts maximum grasp success probability because of the conditioning step that is shown in Figure 4.11. The reason for the more defined plateau in Figure 4.23 is that object pose uncertainty is low enough ($\sigma_{\Delta x_{\text{obj}}} = \sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$) so that there are several grid cells where all classification boundaries overlap, while the immediately affecting base position uncertainty is low ($\sigma_{\Delta x_{\text{rob}}} = 0.00\text{m}$, $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$). Although object pose uncertainty was lower in the top left image of Figure 4.19 ($\sigma_{\Delta x_{\text{obj}}} = 0.00\text{m}$ and $\Delta y_{\text{obj}} = 0.025\text{m}$), the immediately affecting base position uncertainty was higher ($\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$).

The remaining images for $\sigma_{\Delta x_{\text{rob}}} \in \{0.08\text{m}, 0.16\text{m}, 0.24\text{m}, 0.30\text{m}\}$ show that increasing $\sigma_{\Delta x_{\text{rob}}}$ continually lowers maximum grasp success probability and stretches out the overall area of the ARPLACE probability distribution. This is also shown in the left plot of Figure 4.24 that depicts maximum grasp success probability for several values of $\sigma_{\Delta x_{\text{rob}}}$. The right plot depicts the corresponding runtimes which are all close to 44ms.

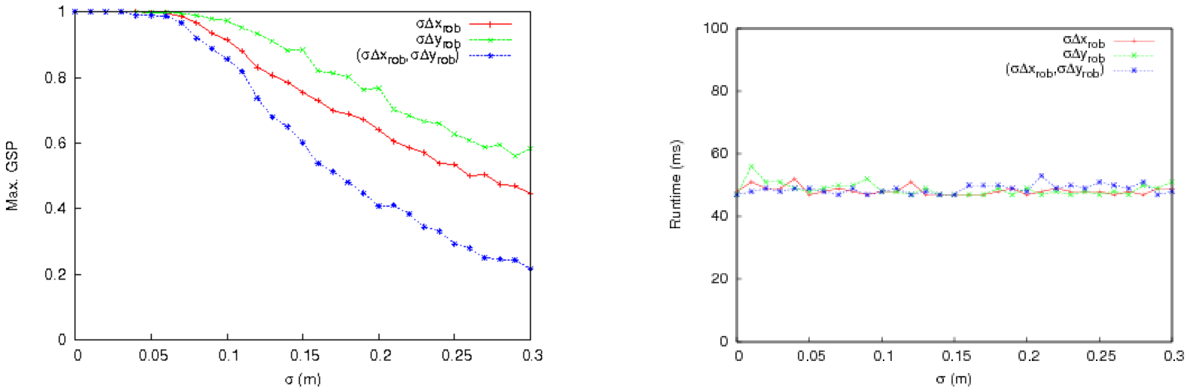


FIGURE 4.24 Maximal grasp success probability (left plot) and runtime (right plot) for $\sigma_{\Delta x_{\text{rob}}} \in \{0.00\text{m}, .., 0.30\text{m}\}$, $\sigma_{\Delta y_{\text{rob}}} \in \{0.00\text{m}, .., 0.30\text{m}\}$, and $(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}}) \in \{(0.00\text{m}, 0.00\text{m}), .., (0.30\text{m}, 0.30\text{m})\}$.

We continue with evaluating the impact of $\sigma_{\Delta y_{\text{rob}}}$ on the ARPLACE probability distribution. The other parameters are set to their default values: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$.

Figure 4.25 depicts ARPLACE probability distributions for the following values of Δy_{rob}

$$\sigma_{\Delta y_{\text{rob}}} \in \{0.00\text{m}, 0.16\text{m}, 0.30\text{m}\}$$

and image of Figure 4.24 depicts maximal grasp success probability and runtime for $\sigma_{\Delta y_{\text{obj}}} \in \{0.00\text{m}, .., 0.30\text{m}\}$.

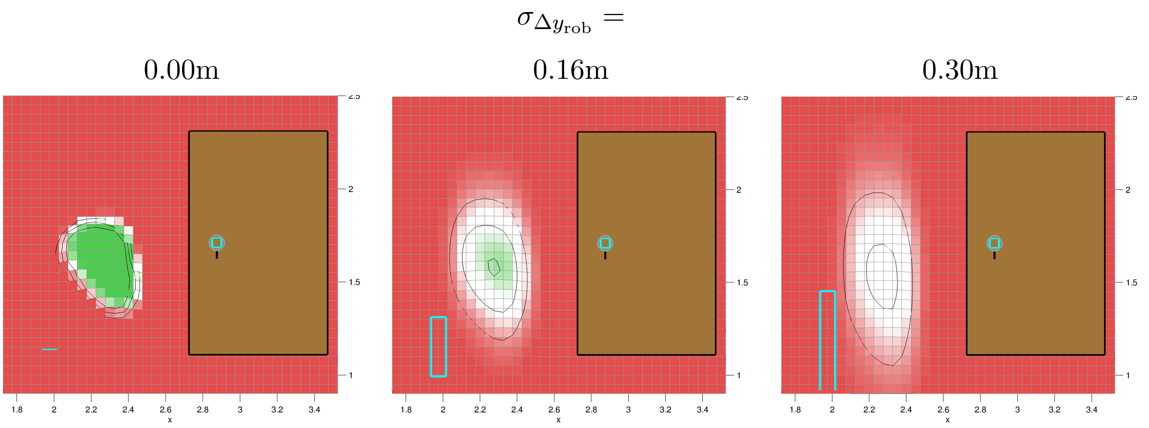


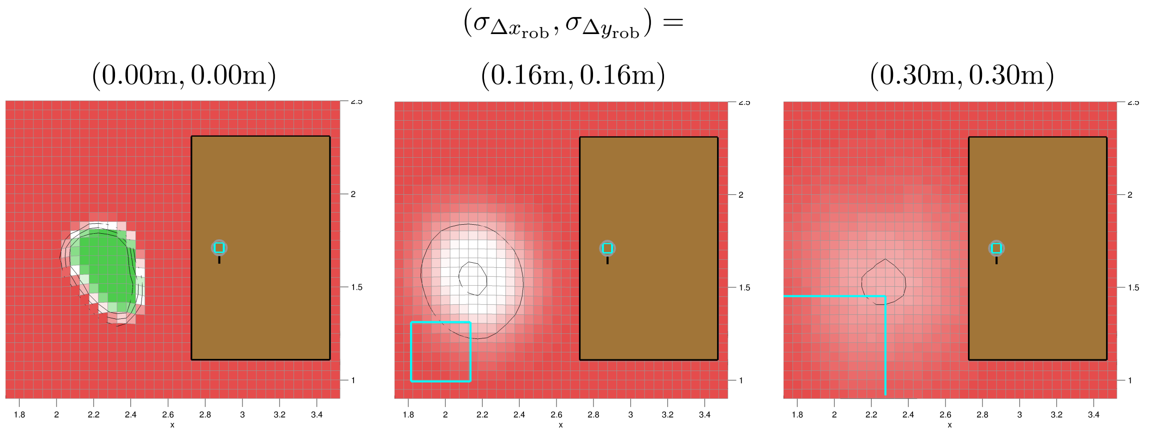
FIGURE 4.25 Visualization of how the ARPLACE changes for different values of $\sigma_{\Delta y_{\text{rob}}}$. The values of $\sigma_{\Delta y_{\text{rob}}}$ are denoted above the plots.

The images show that while $\sigma_{\Delta y_{\text{rob}}}$ is increasing, maximum grasp success probability gets lower, and the overall area of the ARPLACE probability distribution stretches out. The decrease in grasp success probability however is slower when increasing $\sigma_{\Delta y_{\text{rob}}}$ than it is when increasing $\sigma_{\Delta x_{\text{rob}}}$. This indicates that the robot can handle uncertainties of its base position along the table edge better than it can handle uncertainties of the distance between the table and the base position. We assume that the same explanation as in the last section holds: The span width of the robot's arms to the front is limited to 0.84m while the span width to both sides is 1.68m.

To conclude this section, we evaluated the impact of varying $\sigma_{\Delta x_{\text{rob}}}$ and $\sigma_{\Delta y_{\text{rob}}}$ simultaneously. Figure 4.22 depicts ARPLACE probability distributions for the following values of $(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}})$

$$(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}}) \in \{(0.00\text{m}, 0.00\text{m}), (0.16\text{m}, 0.16\text{m}), (0.30\text{m}, 0.30\text{m})\}$$

Because the condition $\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}}$ holds for the plots, the ARPLACE probability distribution shares strong similarities with bivariate Gaussian distributions when base position uncertainty is the dominant factor. This is the case for the center and right plot of Figure 4.22. Figure 4.20 depicts maximal grasp success probability and runtime for $(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}}) \in \{(0.00\text{m}, 0.00\text{m}), .., (0.30\text{m}, 0.30\text{m})\}$.



$(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}}) =$

$(0.00\text{m}, 0.00\text{m})$ $(0.16\text{m}, 0.16\text{m})$ $(0.30\text{m}, 0.30\text{m})$

FIGURE 4.26 Visualization of how the ARPLACE changes for different values of $\sigma_{\Delta x_{\text{rob}}}$ and $\sigma_{\Delta y_{\text{rob}}}$. The values of $(\sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}})$ are denoted above the plots.

4.4 ARPLACE Information for High-Level Planning

This section describes that Action-Related Places provide valuable information for high-level planning systems. This information can be used in order to forestall which actions to perform in order to raise the chances of successful manipulation. In section 4.4.1 we will examine what high-level planning systems can do in order to raise manipulation chances when the ARPLACE reports low maximum grasp success probability. How high-level planning systems can find the reason when a manipulation action failed that was predicted to be successful with high probability is discussed in section 4.4.2.

4.4.1 Raising Manipulation Chances

When an ARPLACE finds very promising manipulation places with grasp success probability of more than 90% there is mostly no need for high-level planning systems to take further actions. But consider the ARPLACE probability distributions in Figure 4.27 for example that were taken from the evaluation in the last section. The high-level planning system can not honestly assume to successfully perform manipulation actions in any of these situations. And although the Action-Related Places look similar to a certain degree, the corresponding situations should be handled differently. A high-level planner that is able to interpret the information that is represented by an ARPLACE can quickly decide which actions will help to improve the situation.

Consider the situation in the left image of Figure 4.27, which was computed for the following task relevant parameters: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.16\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.16\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$. The

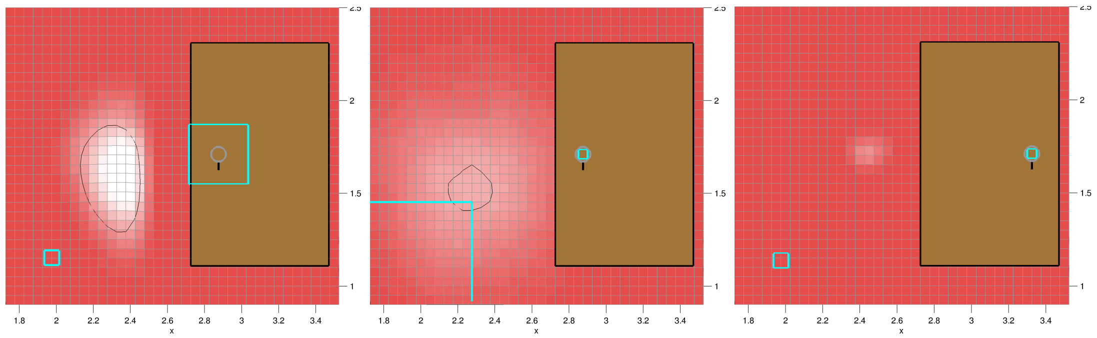


FIGURE 4.27 ARPLACE probability distributions where the robot can not expect to successfully perform the manipulation action. The high-level planner should consider additional actions in order to increase grasp success probability. Left: $\sigma_{\Delta x_{\text{obj}}} = \sigma_{\Delta y_{\text{obj}}} = 0.16m$. Center: $\sigma_{\Delta x_{\text{rob}}} = \sigma_{\Delta y_{\text{rob}}} = 0.30m$. Right: $\Delta x_{\text{obj}} = 0.60m$.

ARPLACE covers a large area but does reach a maximum grasp success probability of only 48.2%. The high-level planner could try to improve the situation by analysing the reasons why the ARPLACE does not find more promising manipulation places. When examining the task relevant parameters it is obvious that the estimation of the object's pose is very uncertain with $\sigma_{\Delta x_{\text{obj}}} = 0.16\text{m}$ and $\sigma_{\Delta y_{\text{obj}}} = 0.16\text{m}$. Therefore, additional measurements from the the vision system are the primary action for being able to find better manipulation places. This is a perfect example where additional exploration and reducing uncertainty may lead to better manipulation chances.

When examining the ARPLACE in the center image of Figure 4.27 it can be seen that the ARPLACE is diffuse. Maximum grasp success probability is as low as 21.6%. The manipulation task is unlikely to succeed and the high-level planner should not execute it in order not to risk injuries. When analysing the task-relevant parameters for computing the ARPLACE: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.30\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.30\text{m}$. The reason for low grasp success probability is clear. The robot's uncertainty into its base position is extremely high with $\sigma_{\Delta x_{\text{rob}}} = 0.30\text{m}$ and $\sigma_{\Delta y_{\text{rob}}} = 0.30\text{m}$. The primary goal is therefore trying to obtain a better localization. Additional range finder measurements, or moving around in order to find landmarks are appropriate actions.

The ARPLACE on the right image of Figure 4.27 is small and has a maximum grasp success probability of only 7.5%. However, additional exploration will not help in this situation. Considering the task relevant parameters (1) $\Delta x_{\text{obj}} = 0.60\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$ it can be seen that the distance of the object to the table edge is $\Delta x_{\text{obj}} = 0.60\text{m}$. So the robot has to completely stretch out his arm making the reaching motion complicated. Of course there is the possibility of moving around the table and grasping the object with the left hand, but this will be discussed in the next chapter. For now let us assume that the robot wants to grasp the object with the right arm from the nearby table edge. There is not much that the robot can do in this situation in order to improve its chances because low maximum grasp success probability is the consequence of a difficult manipulation task. The important aspect is that a high-level planner can infer this fact by examining the ARPLACE and the task-relevant parameters. Knowing that a task is difficult saves time because the high level planner can immediately focus on the primary question. Is it worth to perform the task at the drawback of a presumable failure, or is aborting the task the better option?

4.4.2 Analysing Unexpected Failure

Even when ARPLACEs are used for positioning the robot's base, it is possible that a manipulation action fails. This can have several reasons.

A) The task is difficult
B) The robot made errors in the state estimation process
C) Effects occured that are not familiar to the robot

Tasks are difficult when the target object is hardly reachable. In the left plot of Figure 4.28 for example, there are several reasons why the task is hard. The target object is far away from the table border and the chair blocks the most promising base positions for grasping (considering obstacles in the ARPLACE framework is discussed in the next chapter). Moreover, the cup's handle requires the robot to approach it from a direction that makes it necessary to completely stretch the arm. And finally the uncertainty into the target object's pose is high with $\sigma_{\Delta x_{\text{obj}}} = \sigma_{\Delta y_{\text{obj}}} = 0.1\text{m}$. That is why the most promising base positions reach a grasp success probability of just above 20%. The robot could try to solve this situation by either trying to succeed, aborting the task, or asking a human person for help. However, when performing the task and failing, then the high-level planning system knows the reason for failing. It was a difficult task because of circumstances like blocking obstacles, huge distances to the manipulation place, or awkward object orientation. Failing had to be taken into account before performing the manipulation.

In case B) the robot failed because it made assumptions that do not hold. Maybe the robot overlooked an obstacle, or was too certain about the position of the target object. Please note that high state estimation uncertainty and erroneous state estimation are not the same.

Erroneous state estimation is not observed by the robot and leads to *wrong* ARPLACEs. The robot will perform the manipulation action in the face of misleading certainty. The center

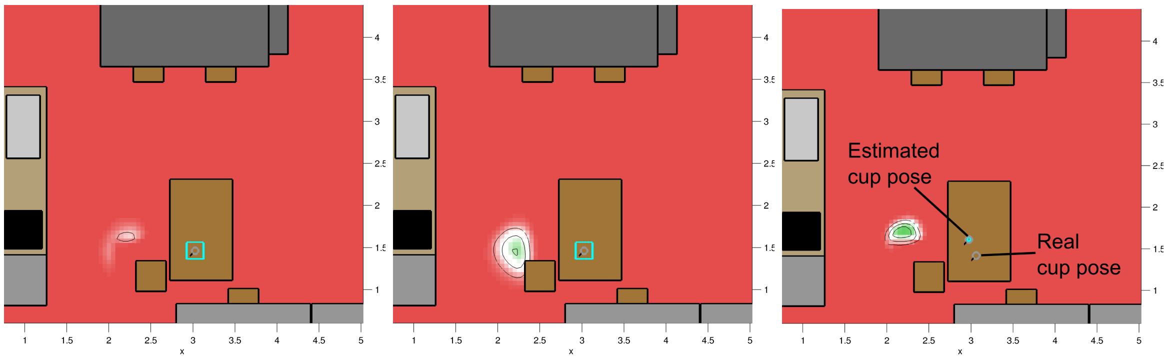


FIGURE 4.28 Reasons why a manipulation action may fail. Left: The grasping task is difficult. Center: The robot overlooked the chair and will bump into it. Right: The robot estimates the cup to be at a wrong pose.

plot of Figure 4.28 depicts a scenario where the robot overlooked the chair at the kitchen table. As a result, the area of promising grid cells that is occupied by the chair was not erased. The robot will navigate towards the region with grasp success probability of more than 80% and bump into the chair. The right plot of Figure 4.28 depicts a scenario where the robot is too certain about its estimation. The robot predicts the object to be located at the wrong pose and additionally being inappropriately certain that his estimation is right. The consequence is that the robot computes a wrong ARPLACE.

High state estimation uncertainty on the other side leads to an ARPLACE with low grasp success probability, but an ARPLACE that is *correct*. The left and center image of Figure 4.27 are examples for ARPLACEs with high state estimation uncertainty. Because the robot is aware of high uncertainty the success probability can be improved by performing additional exploration.

In case C) the robot fails because of reasons that have an impact on the manipulation action but are not considered by the robot. A manipulator joint might have been exchanged lately which leads to lower friction, or a new motor controller has been added. These are robot skills that are encoded in Generalized Success Models. Significant changes in the robot's skills therefore requires the robot to re-learn Generalized Success Models, while small gradual changes like an increase in joint friction over time may be included by continually updating the Generalized Success Model. However, updating Generalized Success Models online is not done yet, but would be a tempting topic for future research. To sum up, if the robot's skills change but this is not reflected in the Generalized Success Model, then the computed ARPLACE probability distribution will overestimate grasp success probability.

The question is how high-level planning systems should react if a manipulation action fails. When maximum grasp success probability is low, we already discussed that the high-level planning system can infer the problematic aspects of the manipulation task. In case state estimation uncertainties are low, then the task is either difficult or additional exploration is required. This can mean to either advise the vision system to predict the target object's pose more precisely, or to make the localization system find out more about the current base position. Whatever might be the reasons, the high-level planning system can infer them and act rationally.

The hard problems are those where the ARPLACE predicts a grasping action to succeed with high probability, but it fails nevertheless. In this case the high-level planning system has to guess about the reasons. The robot could learn new Generalized Success Models in order to reflect changed robot skills or perform additional exploration. However, the high-level planning system is not certain which action might improve the situation. Additionally, updating Generalized Success Models requires a significant amount of time and should be delayed until idle periods.

4.5 Results from the Simulated Robot

In this section we evaluate if manipulation places that are proposed by the ARPLACE framework are superior to other place-finding strategies. In this evaluation, the position to which the robot navigates is the position for which the ARPLACE framework computed the highest probability that grasping the target object will succeed. We call this ARPLACE-based navigation strategy. We compare this strategy to a place-finding strategy that is called FIXED, which always navigates to a location that has the same relative offset to the target object, whilst at the same time taking care not to bump into the table. FIXED chooses the manipulation pose by trying to keep a distance of $\Delta x_{\text{rob}} + \Delta x_{\text{obj}} = 0.68\text{m}$ and $\Delta y_{\text{rob}} = -0.09\text{m}$ between its base position and the target object. If the robot would bump into the table, because the target object is too far away from the table edge, then the robot approaches the table as close as possible in order to try to keep the distance between its shoulder and the target object small.

The left image of Figure 4.29 depicts an example where the cup's distance to the table edge is 0.3m. The robot therefore chooses to keep a distance of 0.38m to the table, and sets $\Delta y_{\text{rob}} = -0.09\text{m}$. In the right image of Figure 4.29 $\Delta x_{\text{obj}} = 0.5\text{m}$. Because the robot wants to keep a distance of at least 0.25m in order not to bump into the table, it chooses the minimum value of $\Delta x_{\text{rob}} = 0.25\text{m}$ and the default value of $\Delta y_{\text{rob}} = -0.09\text{m}$.

The reason why we chose the FIXED navigation strategy for comparing it to the ARPLACE navigation strategy is that FIXED achieved the highest number of successful manipulation actions among all navigation strategies that propose to move to a specific offset with respect

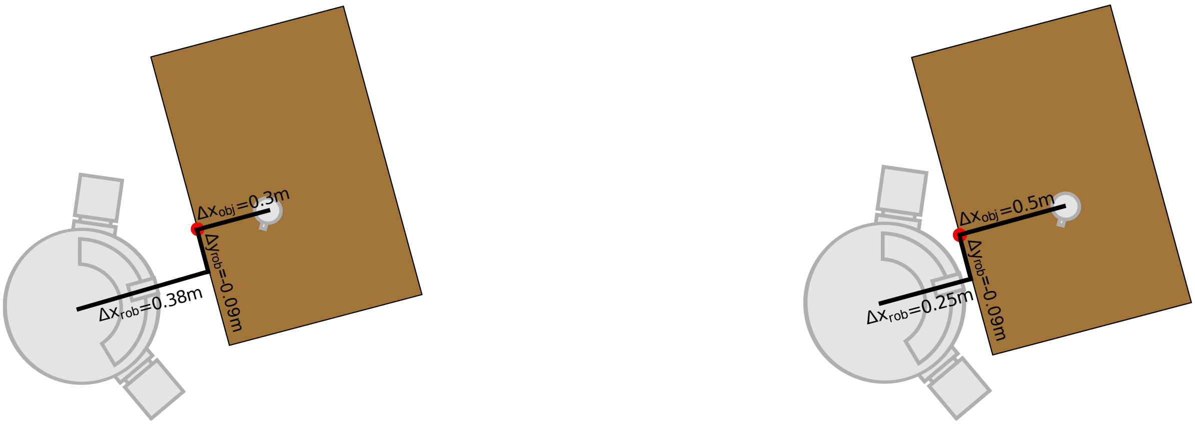


FIGURE 4.29 Manipulation places that are proposed by the FIXED navigation strategy. The origin of the relative feature space is colored red.

to the target object. That is why FIXED is the intuitive benchmark navigation strategy.

In the experiments, we vary the uncertainty of the robot into its base position $\langle \sigma_{\Delta x_{\text{rob}}}, \sigma_{\Delta y_{\text{rob}}} \rangle$, and into the pose of the cup $\langle \sigma_{\Delta x_{\text{obj}}}, \sigma_{y_{\text{obj}}} \rangle$. For each combination of these parameters, the robot performs navigate-reach-grasp-lift sequences and records the result just as during data acquisition for learning the Generalized Success Model. To simulate the uncertainty, we sample a specific 'perceived' robot and cup position from the probability distribution that is defined by their mean and covariance matrix. The result of the action is determined by the true simulated state of the world, but the robot grounds its decisions in the perceived samples.

The results of this evaluation are summarized in the three bar plots in Figure 4.30, which depict the success ratios of the ARPLACE-based and FIXED navigation strategies. Each ratio is computed from over 100 examples. We performed a hypothesis test for the following hypothesis: "when performing a manipulation action from manipulation places that are proposed by the ARPLACE framework, then the probability of successful grasping is significantly higher than when performing a manipulation action from places that are proposed by the FIXED strategy". The p-value above each pair of bars is computed with a χ^2 test in order to support this hypothesis.

The first graph depicts the success ratios for increasing uncertainty about the object pose ($\sigma_{\Delta x_{\text{obj}}} \in \{0.00\text{m}, 0.05\text{m}, 0.75\text{m}, 0.10\text{m}, 0.15\text{m}, 0.20\text{m}\}$) when robot position uncertainty is set to $\sigma_{\Delta x_{\text{rob}}} = 0.05\text{m}$. In all cases, the ARPLACE strategy significantly outperforms the FIXED strategy ($p < 0.01$). Furthermore, the performance of ARPLACE is much more robust towards increasing uncertainty of object pose. For example, when the uncertainty into the object's pose $\sigma_{\Delta x_{\text{obj}}} = 0.15\text{m}$ then manipulation was successful 90% of the time when performed from places that were proposed by the ARPLACE framework. Manipulation was successful

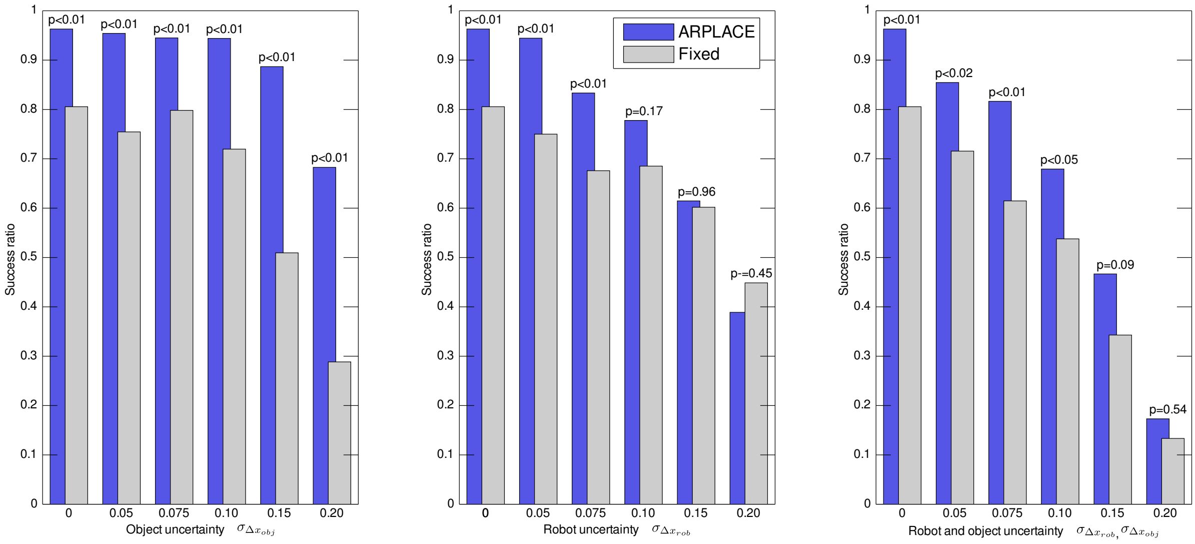


FIGURE 4.30 Success ratios of the ARPLACE and FIXED navigation strategies when changing the uncertainties into object pose (left imaag), robot position (center image), and robot and object position (right image).

only 50% of the time, when performed from places that were proposed by the FIXED strategy. This indicates that the design goal of making the ARPLACE framework robust against state estimation uncertainties into the object's pose was accomplished.

The same trend can be observed when increasing the uncertainty of the robot into its base position from 0.00m to 0.20m and setting object pose uncertainty to $\sigma_{\Delta x_{\text{obj}}} = 0.05\text{m}$. It can be seen that manipulation places proposed by the ARPLACE framework are significantly superior when $\sigma_{\Delta x_{\text{rob}}} < 0.10\text{m}$. However, when $\sigma_{\Delta x_{\text{rob}}} > 0.10\text{m}$, then the difference between ARPLACE and FIXED is no longer significant. The last graph depicts the success ratios when increasing both robot and object uncertainty. Again, ARPLACE significantly outperforms FIXED.

Summarizing, ARPLACE is more robust towards state-estimation uncertainties than the benchmark navigation strategy FIXED. The effect is more pronounced for uncertainties into object pose than it is for uncertainties into robot position.

CHAPTER 5

Refining Action-Related Places

The last chapter described how Action-Related Places are computed online by estimating the robot's base position and the target object's pose, and using these estimations as input for querying the Generalized Success Model. It was explained how Monte-Carlo simulation and conditioning techniques are used to derive an Action-Related Place that is based on grasp success probability. In order to focus on the basic ARPLACE algorithm we constrained the considered task to grasping cups from a specific table edge. In this chapter we will extend the concept of Action-Related Places and present how ARPLACEs are refined in order to make them more general.

Figure 5.1 depicts an overview of the refinement techniques that are explored in this chapter and depicts exemplary ARPLACE plots that illustrate the corresponding refinement technique.

The rest of this chapter is structured as follows. Section 5.1 presents related work. Several ARPLACEs for grasping from the top are depicted in section 5.2, and it is analyzed what impact the varying of task-relevant parameters has on the resulting ARPLACE probability distribution. Section 5.3 introduces multi modal ARPLACE probability distributions and section 5.4 describes how ARPLACEs for grasping with the left arm can be computed without learning an additional Generalized Success Model. Section 5.5 deals with the question how obstacles can be taken into account. Section 5.6 investigates how ARPLACEs for grasping multiple objects from a single base position are computed, and presents how a transformational planning system decides whether grasping multiple objects at once is advantageous or not. How the ARPLACE framework accounts for uncertainty into the target object's type is studied in section 5.7, and section 5.8 describes how to refine ARPLACEs when grasping from a promising manipulation place was not successful in the first attempt.

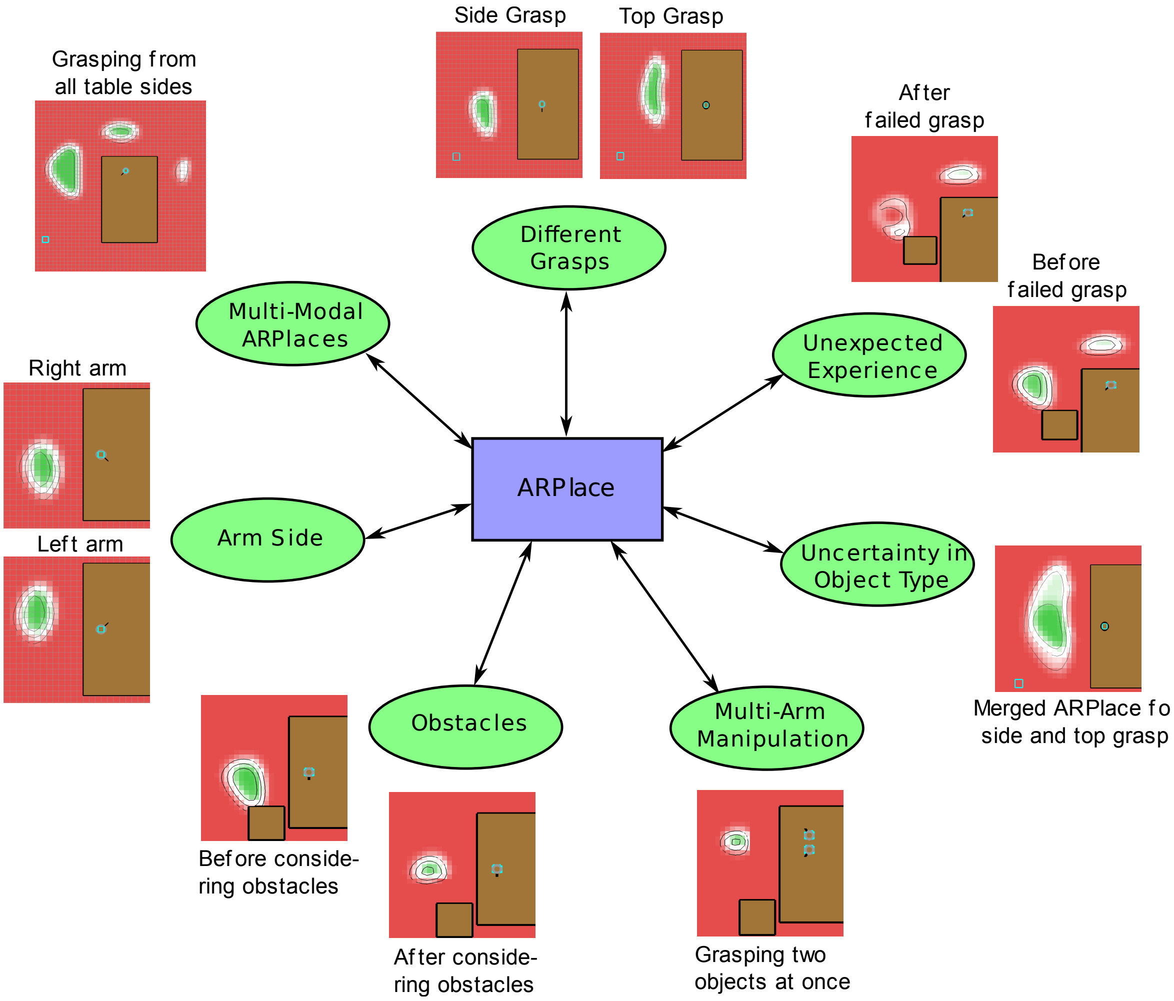

Side Grasp
Top Grasp
Grasping from all table sides
After failed grasp
Before failed grasp
Different Grasps
Multi-Modal ARPlaces
Unexpected Experience
Right arm
Left arm
ARPlace
Arm Side
Uncertainty in Object Type
Merged ARPlace fo side and top grasp
Obstacles
Multi-Arm Manipulation
Before conside-ring obstacles
After conside-ring obstacles
Grasping two objects at once

FIGURE 5.1 Visual overview of chapter 5. Green circles depict refinement techniques. ARPLACE plots next to refinement techniques visualize an examplary application of the corresponding refinement technique.

5.1 Related Work

Friedman and Weld (1996) demonstrate the advantages of least-commitment planning. They showed that setting open conditions to abstract actions and later refining this choice to a particular concrete action can lead to exponential savings. The principle of lazy evaluation was applied to motion planning by Bohlin and Kavraki (2000). They were able to significantly reduce the number of collision checks for building probabilistic roadmaps.

Sussman (1973) was the first to realize that *bugs* in plans do not only lead to failure, but are actually a source of information to construct improved and more robust plans. Although this research was done in the highly abstract symbolic blocks world domain, this idea is still fundamental to transformational planning.

Beetz (2001) described the declarative and expressive plan language RPL that is the basis of our transformational planning system. The constraints for plan design, especially the specification of declarative goals that indicate the purpose of code parts, have been shown by Beetz and McDermott (1992). Our system scales with respect to the modeling of navigation tasks, and to reasoning about perception that is based on computer vision, the relation between objects and their representation in the robot's belief, as well as reasoning about complex manipulation tasks.

Temporal projection is an integral component of a transformational planning system. McDermott (1997) developed a powerful, totally ordered projection algorithm that is capable of representing and projecting various kinds of uncertainty, concurrent threads of execution, and exogenous events. Beetz and Grosskreutz (2000) extended the language for specifying action models and grounded their representation into probabilistic hybrid automata as a formal underpinning. The representation language was shown to be rich enough to accurately predict reactive navigation behavior of an autonomous robot office courier.

5.2 ARPLACEs for Grasping from the Top

ARPLACE probability distributions for grasping objects from the top are computed just like ARPLACEs for grasping from the side. That is why the algorithm for computing ARPLACEs that was described in section 4.2 also applies here. However, ARPLACEs for grasping from the side and ARPLACEs for grasping from the top are not identical because the Generalized Success Model that is used for reconstructing classification boundaries is different.

This section presents several examples for ARPLACEs for grasping objects from the top. 24.255 experiments were used to learn the corresponding Generalized Success Model as was described in section 3.6.2. Figure 5.2 depicts an ARPLACE for grasping a glass from the top. The following task relevant parameters were used: (1) $\Delta x_{\text{obj}} = 0.15\text{m}$; (2) $y^{\text{tab}}_{\text{obj}} = 0.00\text{m}$; (3) $\Delta\psi_{\text{obj}} = -\frac{\pi}{2}$; (4) $\sigma_{\Delta x_{\text{obj}}} = 0.025\text{m}$; (5) $\sigma_{\Delta y_{\text{obj}}} = 0.025\text{m}$; (6) $\sigma_{\Delta x_{\text{rob}}} = 0.04\text{m}$; (7) $\sigma_{\Delta y_{\text{rob}}} = 0.04\text{m}$.

In the following, we will briefly examine the effect that varying each task-relevant parameter has on the ARPLACE. Figure 5.3 depicts how the ARPLACE changes when a certain task-relevant parameter is varied. One task-relevant parameter is changed in the first five plots, while the other six parameters are set to their default value. In the last plot on the bottom right, Δx_{rob} and Δy_{rob} are jointly varied.

The following results can be observed.

- Top left image: When the object's distance to the table edge increases, the ARPLACE

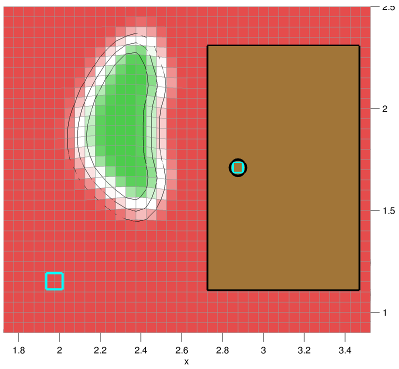


FIGURE 5.2 Visualization of an ARPLACE for grasping a glass from the top.

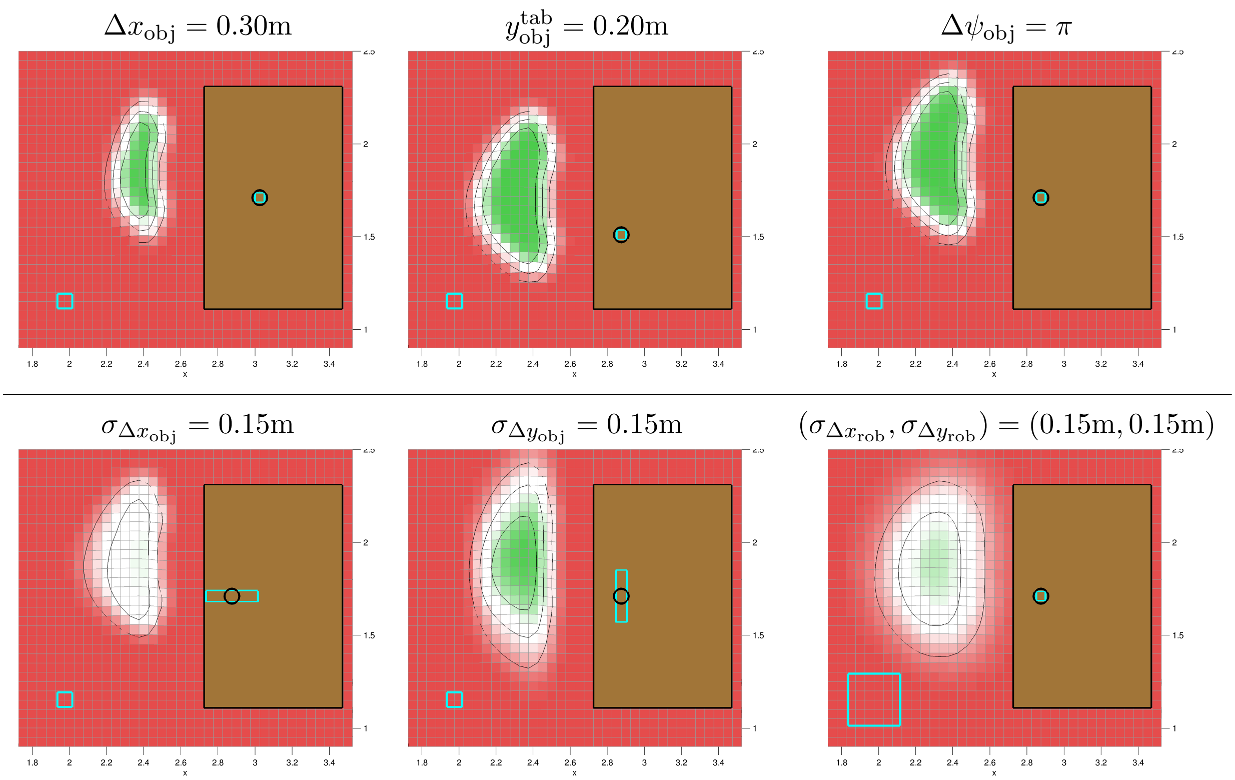
moves closer to the table, and the overall area of promising base positions shrinks.

- Top center image: When the object moves along the table edge, the ARPLACE follows by the same distance without changing its shape
- Top right image: When the angle of the object changes, the ARPLACE remains the same
- Bottom left image: When the uncertainty into the object's distance to the table edge increases, grasp success probability decreases while the contours of the ARPLACE remain
- Bottom center image: When the uncertainty into the object's position along the table edge changes, grasp success probability decreases and the ARPLACE stretches out more along the table edge
- Bottom right image: When the uncertainty into the robot's base position increases, the area of the ARPLACE gets bigger and grasp success probability decreases

The results are similar to the results for grasping from the side. One major difference remains. Changing the object's orientation does not have an impact on the ARPLACE probability distribution.

5.3 Multi Modal ARPLACEs

Until now we considered grasping the target object from the left table edge. However, grasping from any table edge has to be possible. The solution is straightforward because of the relative feature space. The robot estimates the target object's pose in world coordinates, and then computes the object's pose relative to all table edges. This enables the robot to compute an ARPLACE probability distribution for every table edge. The top left image of Figure 5.4 depicts a scenario where the robot estimates the cup's pose to be $\langle 3.02\mathrm{m}, 2.1\mathrm{m}, -\frac{3}{4}\pi\rangle$ relative


$\Delta x_{\mathrm{obj}} = 0.30\mathrm{m}$

$y^{\mathrm{tab}}_{\mathrm{obj}} = 0.20\mathrm{m}$

$\Delta\psi_{\mathrm{obj}} = \pi$

$\sigma_{\Delta x_{\mathrm{obj}}} = 0.15\mathrm{m}$

$\sigma_{\Delta y_{\mathrm{obj}}} = 0.15\mathrm{m}$

$(\sigma_{\Delta x_{\mathrm{rob}}}, \sigma_{\Delta y_{\mathrm{rob}}}) = (0.15\mathrm{m}, 0.15\mathrm{m})$

FIGURE 5.3 Visualization of the impact of changing task-relevant parameter on the ARPLACE that is depicted in Figure 5.2. The task-relevant parameter(s) that change is depicted above the subplot. All other parameters are set to their default value.

to the world frame. Subsequent images show the cup's pose relative to every table edge and the corresponding ARPLACEs. It can be seen that the cup can be grasped from the left and top table edge.

The resulting ARPLACE is computed by merging the ARPLACEs for each table edge. The merging is done by using the max-operator as follows

$$p(x, y) = max(p_i(x, y)) \; ; \; i \in 1, .., n \tag{5.1}$$

where $p_i(x, y)$ is grasp success probability of grid cell (x, y) when computing an ARPLACE for tabel edge i. n is the overall number of table edges. The resulting ARPLACE is depicted in the bottom right image of Figure 5.4. It can be seen that a multi modal ARPLACE probability distribution emerges.

Because an ARPLACE has to be computed for every table edge the runtime for computing multi-modal ARPLACEs increases accordingly. If the target object is reachable from every table edge, then worst case runtimes of 180ms are required. The runtime for computing the

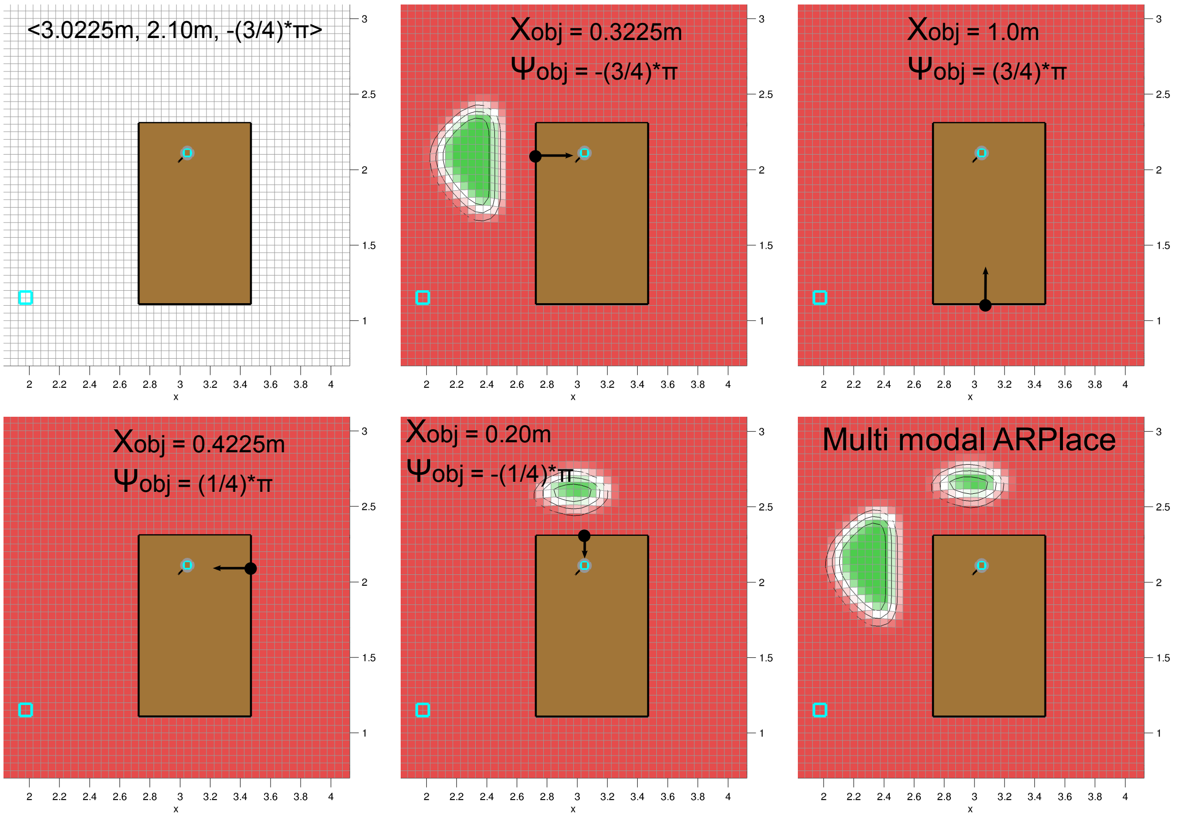


FIGURE 5.4 Top left: Scenario with the robot's estimation of the cup's absolute pose. The other plots depict ARPLACEs for grasping the cup from the left table edge (top center plot), from the bottom table edge (top right plot), from the right table edge (bottom left plot), and from the top table edge (bottom center plot). In every plot, the cup's pose with respect to the relative feature space (Δx_{obj} and $\Delta\psi_{\mathrm{obj}}$), the origin of the relative feature space (big black dot), and the x-axis of the relative feature space (black arrow) is drawn. Bottom right: Multi modal ARPLACE for grasping from any table edge.

ARPLACEs in the above example was 89ms. 45ms for the ARPLACE from the left table edge, 43ms for the ARPLACE from the top table edge, and 1ms for the ARPLACEs of the bottom and right table edge. Because the computation of different ARPLACEs share no data, computing multi-modal ARPLACEs scales very well when it is distributed among multiple CPUs. The step of merging all ARPLACEs with the max operator took only 0.14ms.

5.4 ARPLACEs for Grasping with the Left Arm

The Generalized Success Models that we presented in chapter 3 were learned for grasping with the right arm. However, computing manipulation places for grasping with the left arm has to be supported by the ARPLACE framework. In the general case an additional Generalized Success

Model needs to be learned for grasping with the left arm. Learning an additional Generalized Success Model and using it in order to compute ARPlaces is straightforward. However, this approach requires additional time in the offline phase because we have to perform additional experiments.

If the kinematics of the robot's arms have the property of being mirror images of each other, then a Generalized Success Model that was learned for one arm can be used for computing ARPlaces for both arms. Figure A.3 depicts the arm kinematics of the Powercube arms. It can be seen that the coordinate frames were chosen so that one arm is a mirror image of the other arm. When the corresponding joints of both arms are turned to the same angle, each arm moves to the mirrored direction of the other arm. Figure 5.5 depicts examples.

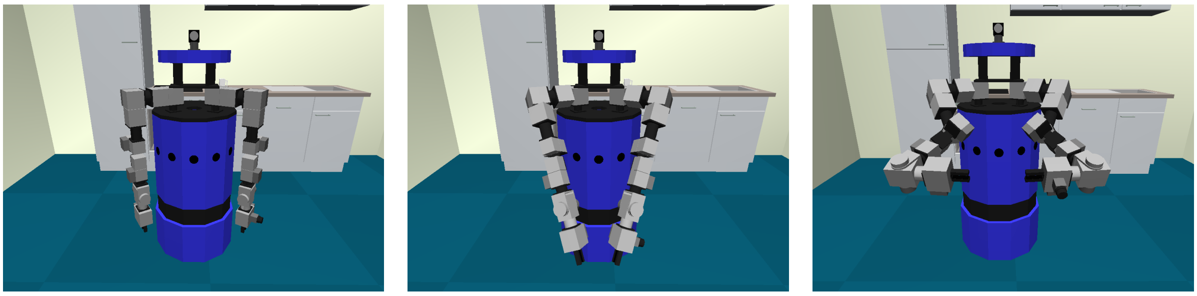


Figure 5.5 Different manipulator configurations show that the kinematics of the Powercube arms are mirror images of each other. The joint angles in the following arm configurations are specified in radians. Left image: Home pose with a configuration of $\langle 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\rangle$ for both arms. Center image: First two joints are rotated. The configuration of both arms is $\langle 0.6, 0.8, 0.0, 0.0, 0.0, 0.0\rangle$. Right image: Both arms are set to configuration $\langle 0.8, 3.0, -1.0, -2.6, -2.0, 0.0\rangle$.

The mirror property of manipulator kinematics makes it possible to use the Generalized Success Model that was learned for grasping with the right arm to compute ARPlaces for grasping with the right *and* left arm. Figure 5.6 depicts an ARPlace for grasping the cup with the right arm.

It can be seen that there is no grid cell where grasp success probability is above 0%, so grasping with the right arm seems to be impossible. This is intuitive because the cup is oriented in a way that would require the right arm to be extremely rotated, which is beyond the manipulator's joint limits. However, when observing the scenario, then a grasp with the left arm seems to be possible.

Figure 5.7 visualizes the computational process for computing the ARPlace for grasping with the left arm. The top left image depicts the scenario. The first step is to find the equivalent problem for grasping with the right arm. Because the kinematics of the manipulators are mirror images of each other, the equivalent problem can be found by mirroring the cup's pose with

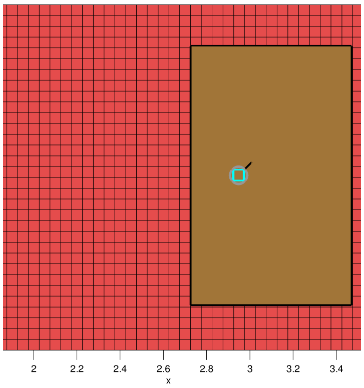


FIGURE 5.6 ARPLACE for grasping the cup with the right arm.

respect to an axis that is normal to the table edge and goes through the center of the cup. The result of the mirroring can be seen in the top right image of Figure 5.7. The pose of the cup with respect to the relative feature space of the left table edge changes from $\langle 0.15\mathrm{m}, \frac{\pi}{4}\rangle$ to $\langle 0.15\mathrm{m}, -\frac{\pi}{4}\rangle$. The second step is to compute the ARPLACE probability distribution for grasping with the right arm. The result can be seen in the bottom right image of Figure 5.7. The third step is to mirror the ARPLACE with respect to the mirror axis that has already been used in step 1. The resulting ARPLACE for grasping the cup with the left arm is depicted in the bottom left image of Figure 5.7.

The runtimes for the three steps that are depicted in Figure 5.7 are below 0.1ms for mirroring object pose, 45ms for computing the ARPLACE, and below 0.1ms for mirroring the ARPLACE.

When we apply the extended algorithm to the example of multi modal ARPLACEs from the last section, we recognize that the cup is not only graspable with the right arm from the left and top table edge, but can also be grasped with the left arm from the right table edge. The resulting ARPLACE probability distribution is depicted in Figure 5.8.

5.5 Taking Obstacles into Account

Until now ARPLACE probability distributions mapped grid cells to the probability that the grasping action would succeed. It was not checked if grid cells are blocked by obstacles. Even base positions with a grasp success probability of 1.0 are meaningless when they are occupied. Clearly, a robot must consider potential obstacles when choosing manipulation places. In this section we explain how obstacles are taken into account by computing a obstacle probability distribution and merge it with grasp success probability.

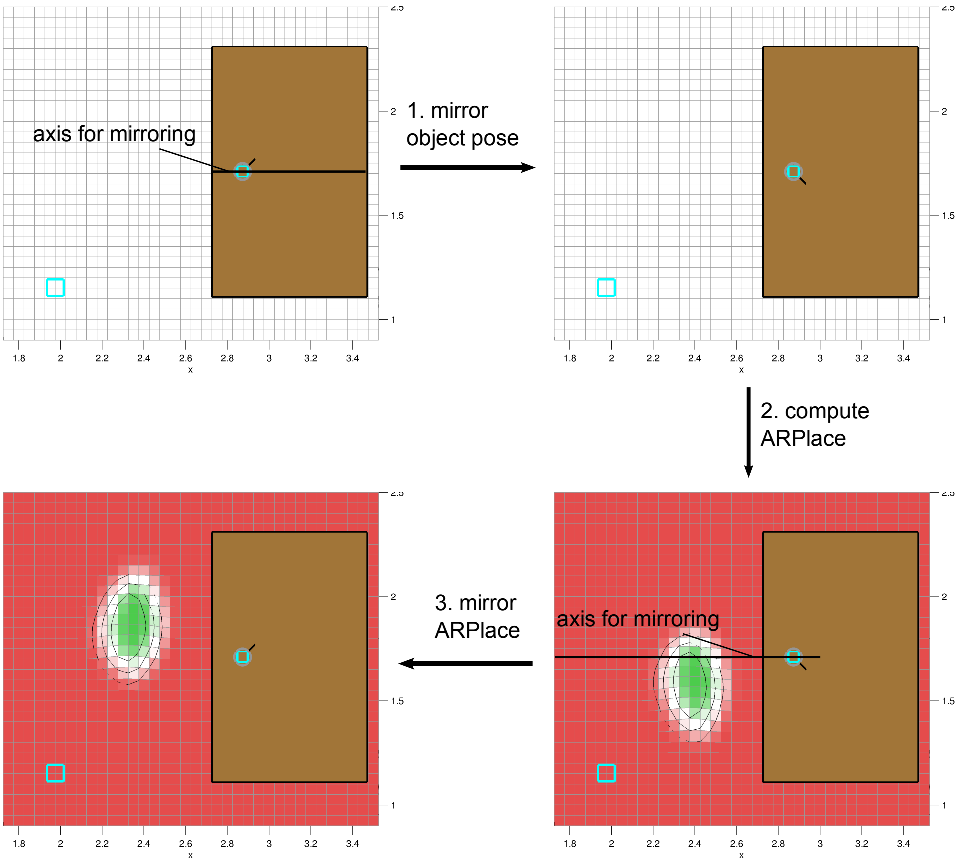


FIGURE 5.7 Top left: Scenario where cup should be grasped with left arm. Top right: Object pose was mirrored. Bottom right: ARPLACE for grasping the mirrored cup with the right arm. Bottom left: ARPLACE for grasping the cup with the left arm.

5.5.1 Obstacle Probability Distribution

A obstacle probability distribution maps grid cells to the probability that it is occupied by an obstacle. It is therefore a mapping

$p_O(x, y) \rightarrow [0; 1]; x \in \mathbb{Q}, y \in \mathbb{Q}$

where x and y refer to the grid cell's index along the x- and y-axis. The algorithm for discretizing space into grid cells is the same as the one for discretizing space into grid cells for computing grasp success probability. As a result, the grid cells of grasp success probability and obstacle probability are aligned.

The left image of Figure 5.9 depicts a scenario where the robot wants to grasp a cup on the table. To compute the obstacle probability distribution, the robot has to identify obstacles in its environment. The center image of Figure 5.9 depicts the (boolean) obstacle probability distribution when the robot has access to ground truth data. Every grid cell (x, y) that is known to be occupied by an obstacle is colored red, indicating that the grid cell is occupied with

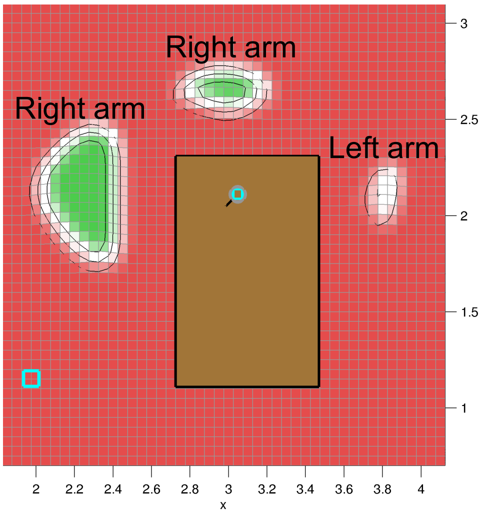


FIGURE 5.8 Revisiting the multi modal ARPLACE that was depicted in the bottom right image of Figure 5.4. The robot now considers grasping with the left arm. Because this is possible from the right table edge, a third cluster of promising grid cells emerges. The label above each ARPLACE cluster denotes which arm side has to be used when performing a grasp from grid cells within the cluster.

probability $p_O(x, y) = 1.0$. A grid cell is considered to be occupied if an obstacle covers it, even if the coverage is partial. Green grid cells are known to be free and have a occupancy probability of 0%. When obstacle shapes of multiple objects overlap, like the kitchen table and the chair nearby, their probability values $p_O(x, y)$ are added, and capped by a probability value of 1.0. In the center plot of Figure 5.9, adding obstacle probabilities has no effect on the obstacle probability distribution, because all probability values are either 0.0 or 1.0. However, adding obstacle probabilities will have an effect later when we consider uncertainties into obstacle poses and probability values may be between 0.0 and 1.0.

It is obvious that not every grid cell that is green in the center image of Figure 5.9 is reachable, like the area between the kitchen table and the worktable below. The reason is that the robot is not point-like, but has a circular body. To be able to treat the robot as a point, we grow the obstacles by the robot's radius. The result is depicted in the right image of Figure 5.9. It can be seen that the obstacle probability distribution is still boolean, and the obstacle regions of the kitchen table, and the worktables have merged to a single, big obstacle region.

5.5.2 Including State Estimation Uncertainties

In real environments the robot has no access to ground truth data, but uses its vision system to estimate obstacle poses. Mean pose, length, and width are estimated together with a covariance matrix Cov_{obs}

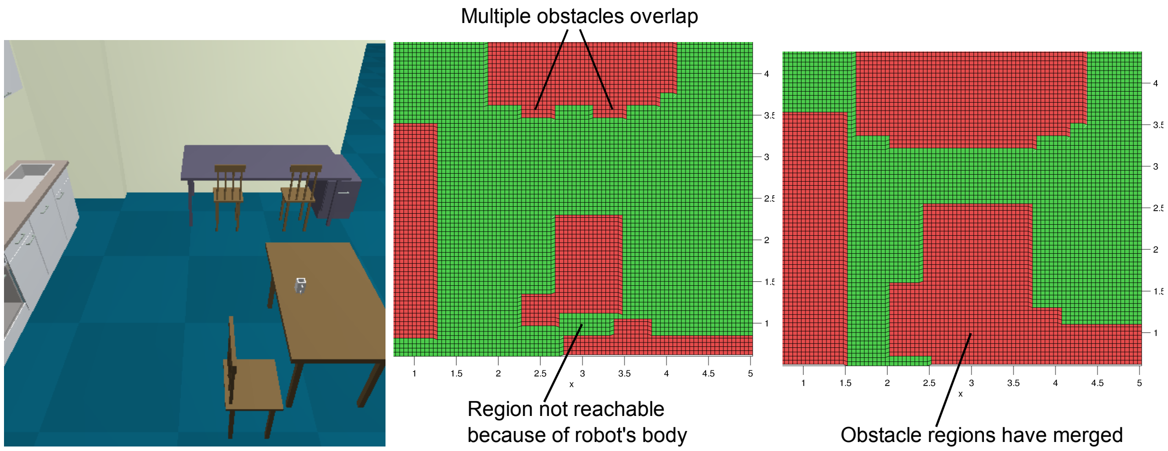


FIGURE 5.9 Left: Kitchen scenario. Center: Obstacle probability distribution for ground truth data. The borders of grid cells are colored black. Grid cells that are colored red inside are known to be occupied and have a occupancy probability of 100%. Green grid cells are known to be free and have a occupancy probability of 0%. Right: Obstacle probability distribution that has been grown by the robot's radius.

$$Cov_{obs} = \begin{pmatrix} Cov(x_{obs}, x_{obs}) & Cov(x_{obs}, y_{obs}) \\ Cov(y_{obs}, x_{obs}) & Cov(y_{obs}, y_{obs}) \end{pmatrix}$$

that reflects the robot's uncertainty into the obstacle's pose. In the following, we compute the standard deviations $\sigma(x_{obs}, x_{obs}) = \sqrt{Cov(x_{obs}, x_{obs})}$ and $\sigma(y_{obs}, y_{obs}) = \sqrt{Cov(y_{obs}, y_{obs})}$ and abbreviate $\sigma(x_{obs}, x_{obs})$ by $\sigma_{x_{obs}}$ and $\sigma(y_{obs}, y_{obs})$ by $\sigma_{y_{obs}}$. In order to take pose uncertainty of obstacles into account, we use the same approach as we did for considering uncertainties into the robot's base position: Selecting random samples from the covariance matrix, summing over these samples to acquire a probability distribution of the obstacle's pose, and conditioning this probability distribution with the object's shape. An example of this process can be seen in Figure 5.10.

The left plot depicts the obstacle probability distribution for the estimated mean pose of a kitchen table. The center plot depicts a probability distribution that was derived by taking 1000 samples from the robot's uncertainty into the table's pose where the positional uncertainty is $\sigma_{x_{table}} = \sigma_{y_{table}} = 0.1\text{m}$.

The probability distribution stretches out to approximately three standard deviations along each axis. The obstacle probability distribution that does consider obstacle pose uncertainty is obtained by conditioning the two above probability distributions. The result is depicted in the right plot of Figure 5.10. It can be seen that many grid cells are not completely green or red any more. Grid cells (x, y) that have a obstacle probability of $p_O(x, y) \in]0, .., 1[$ are colored

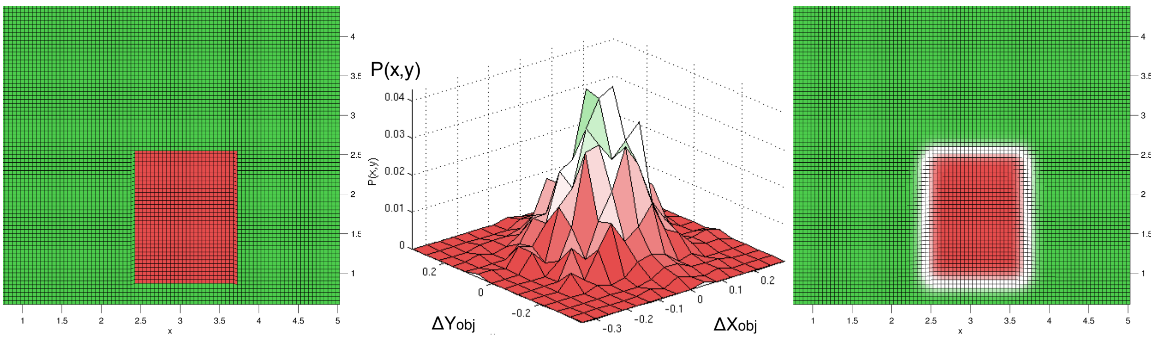


FIGURE 5.10 Taking uncertainty into an obstacle's pose into account. The obstacle we consider here is a kitchen table. Left: Boolean obstacle probability distribution for the table's estimated mean pose. Center: Sampled probability distribution for the uncertainty into the table's pose ($\sigma_{x_{table}} = \sigma_{y_{table}} = 0.1\mathrm{m}$). Right: Obstacle probability distribution for kitchen table when robot's uncertainty into the kitchen table's pose is taken into account. It is the result of conditioning the probability distributions in the left and center plot.

in lighter red, lighter green, or white.

A ARPLACE probability distribution $p(x, y)$ that considers obstacles is computed as follows

$$p(x, y) = p_S(x, y) \cdot p_U(x, y) \tag{5.2}$$

where $p_S(x, y)$ is the grasp success probability that represents the robot's estimated grasp success probability if no obstacles are present. $p_U(x, y)$ is the probability that grid cell (x, y) is *unoccupied*. Because the obstacle probability distribution computes the probability that a grid cell is *occupied* ($p_O(x, y)$), we compute $p_U(x, y)$ as

$$p_U(x, y) = 1 - p_O(x, y) \tag{5.3}$$

In the following, we will refer to an ARPLACE probability distribution $p_S(x, y)$ that *does not* take obstacles into account as an ARPLACE probability distribution that is based on *basic* grasp success probability. An ARPLACE probability distribution $p(x, y)$ that *does* take obstacles into account is referred to as a ARPLACE probability distribution that is based on *combined* grasp success probability.

Probabilistically, equation 5.3 can be read as

$$\begin{aligned} P(\mathit{CombinedSuccess}) &= P(\mathit{Success}, \mathit{Unoccupied}) \\ &= P(\mathit{Success} \mid \mathit{Unoccupied}) \cdot P(\mathit{Unoccupied}) \end{aligned}$$

because $P(\mathit{Success}, \neg\mathit{Unoccupied}) = 0$. It is the application of Bayes' theorem for the combination of conditional probabilities.

Figure 5.11 depicts a more complex example of how the ARPLACE framework takes obstacles into account. The covariance matrices for the kitchen table Cov_{table}, the chair at the kitchen table $Cov_{tabchair}$, chair 1 at the TV table $Cov_{tvchair1}$, and chair 2 at the TV table $Cov_{tvchair2}$ are as follows

$$Cov_{table} = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}, \; Cov_{tabchair} = \begin{pmatrix} 0.0016 & 0 \\ 0 & 0.0016 \end{pmatrix}$$

$$Cov_{tvchair1} = \begin{pmatrix} 0.0016 & 0 \\ 0 & 0.0016 \end{pmatrix}, \; Cov_{tvchair2} = \begin{pmatrix} 0.04 & 0 \\ 0 & 0.04 \end{pmatrix}$$

Ground truth data is assumed for all other objects.

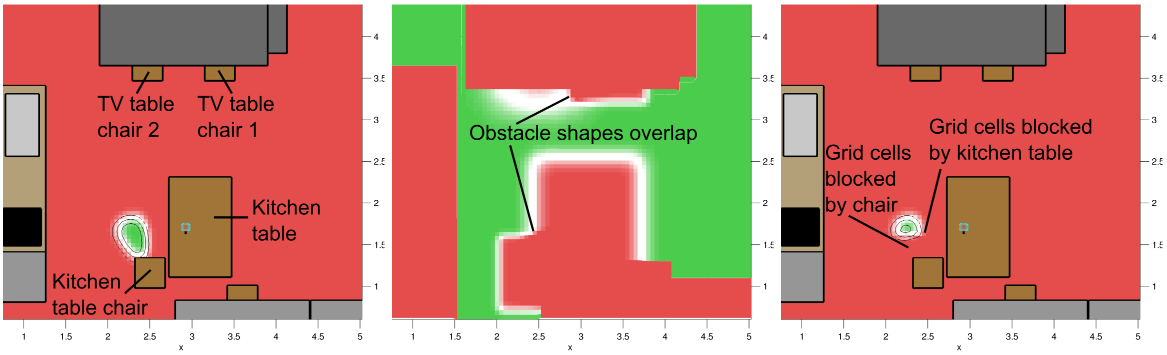


FIGURE 5.11 Merging an ARPLACE and an obstacle probability distribution. Wre omitted grid lines for clarity. Left: ARPLACE for grasping the cup based on basic grasp success probability. Center: Obstacle probability distribution for the robot's current believe state. Right: ARPLACE probability distribution based on combined grasp success probability.

The uncertainty into chair 2 at the TV table is highest. This leads to the result that the shape of this chair in the obstacle probability distribution is blurred the most, followed by the shape of the kitchen table and the other chairs. Furthermore, the obstacle probability shapes of the kitchen table and the chair nearby, as well as the obstacle probability shapes of the chairs at the TV table overlap, and therefore uncertainty values at these grid cells are added as indicated in the center plot of Figure 5.11. Figure 5.11 shows that the area of promising manipulation places in the right plot is significantly smaller than the area of promising manipulation places in the left plot. The reason is that the chair at the kitchen table blocks several grid cells at the bottom of the ARPLACE and the kitchen table blocks some grid cells on the right side.

5.5.3 Performance Analysis

The runtime for computing an obstacle probability distribution was analyzed along two lines. The runtime for computing boolean obstacle probability distributions for the object's mean pose (compare left image of Figure 5.10) is mainly influenced by the number of obstacles and the number of grid cells that are occupied. We created four scenarios to evaluate this. In the first scenario the only obstacle is a chair, and in the second scenario the only obstacle is a kitchen table. The third scenario (called 'small') consisted of two chairs, two tables, and a small cupboard. The fourth scenario (called 'big') consisted of all obstacles in the kitchen scenario that is depicted in the left image of Figure 5.9.

The second parameter that is varied is the number of samples that were used for computing the obstacle's sampled probability distribution that takes state estimation uncertainty of the object's pose into account (compare center image of Figure 5.10). We used 10, 100, and 1000 samples.

The results are depicted in Figure 5.12. Every data point is the mean of 25 experiments. Standard deviations have been computed as well, but were under 10% of the mean value for all experiments. It can be seen that there is a linear relation between the number of samples and the runtime for computing the sampled probability distribution. We found 100 samples to be a good tradeoff between precision and computation time.

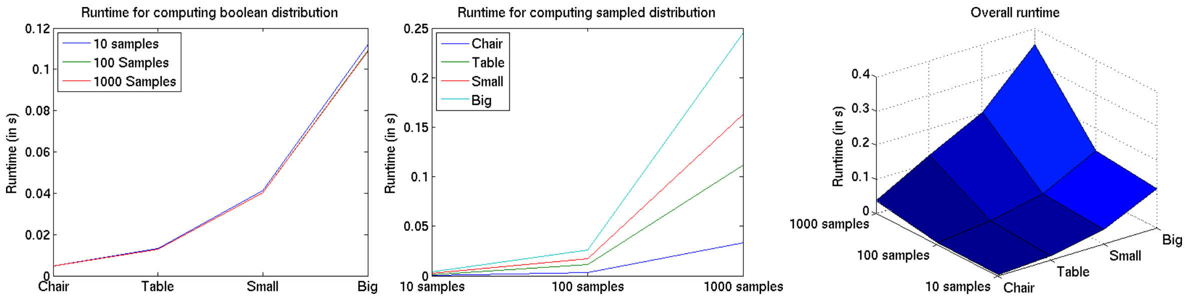


FIGURE 5.12 Runtimes for computing the boolean obstacle distribution for the obstacle's estimated mean pose (left), the sampled probability distribution for taking object uncertainty into account (center), and the overall obstacle probability distribution (right).

The overall computation time of an obstacle probability distribution is almost exclusively determined by the computation times of the boolean obstacle probability distribution and the sampled probability distribution. The merging of these two probability distribution is almost instantaneous at under 0.1ms. For 100 samples, the runtime for computing the obstacle probability distribution is between 8.3ms when there are very few obstacles (scenario 'chair'), and 135.5ms when the environment is full of obstacles (scenario 'big'). We consider this to be sufficiently fast.

Currently the obstacle probability distribution is computed for many grid cells. Even for grid cells from where the object is out of reach. Further runtime improvements could be made if the capability map is used for limiting the obstacle probability distribution to grid cells from where the object is theoretically within reach.

5.6 ARPLACEs for Multi-Arm Manipulation

So far, we have explained how the ARPLACE framework computes the optimal manipulation place for grasping one target object. Subsequently, the robot navigates to the manipulation place and grasps the object. When there are multiple target objects this action-sequence can be applied sequentially. The robot estimates required parameters, decides which object to grasp next, and computes an ARPLACE for the chosen object. When the task is to clean a kitchen table, which means to bring objects from the table to the dishwasher, then this approach requires n grasping actions in order to clean a table with n objects. $2 \cdot n$ navigation actions are required, because for every object the robot has to navigate to the object, grasp it, and then navigate to the dishwasher in order to put it down.

In everyday activities, opportunities for optimizing the course of action arise constantly, as tasks can be interleaved or executed in parallel. For instance, when setting the table, plates can be stacked instead of carrying them one at a time, cupboards can be left open during the task in order not to open them again, or multiple target objects can be grasped from a single base position. Being able to perceive, predict, and exploit such opportunities leads to more efficient and robust behavior. Applied to the above problem of cleaning a table with n objects this could mean that the robot tries to grasp as many objects as possible from a single base position. Our B21r mobile robot has two manipulators and is therefore able to grasp and carry two objects at once. If it is possible to always grasp two objects at once, then the robot requires $2 \cdot \lceil \frac{n}{2} \rceil$ instead of $2 \cdot n$ navigation actions to clean the table. Although the robot has to perform n grasping actions as before, the time for performing the n grasping actions can be reduced because the robot can perform two grasping actions in parallel.

In order to thoroughly exploit possibilities of multi-arm manipulation, the robot must: 1) use least-commitment planning, so not prematurely commit to a specific plan when it is not necessary, because optimization chances may arise during the course of action; 2) have rules for transforming suboptimal plans into more efficient ones. In section 5.6.1, we show how ARPLACE probability distributions are merged with the goal of finding optimal manipulation places for grasping multiple objects at once. Section 5.6.2 evaluates the impact of different object configurations on ARPLACEs for multi-arm manipulation. Section 5.6.3 gives an overview of our high level planning system that evaluates whether performing multiple grasps at once is advantageous, or if the default approach of sequentially grasping objects is more desirable.

5.6.1 Merging ARPLACEs for Multi-Arm Manipulation

Figure 5.13 depicts a scenario with two cups as target objects. The cups' poses are $obj_1 = \langle 3.0\text{m}, 1.85\text{m}, \frac{4}{3}\pi \rangle$ and $obj_2 = \langle 3.0\text{m}, 2.0\text{m}, \frac{2}{3}\pi \rangle$ with respect to the world frame. The uncertainty into the objects' poses are $\sigma_{x_{obj_1}} = \sigma_{y_{obj_1}} = \sigma_{x_{obj_2}} = \sigma_{y_{obj_2}} = 0.04\text{m}$, and the uncertainty into the robot's base position is $\sigma_{x_{\text{rob}}} = \sigma_{y_{\text{rob}}} = 0.03\text{m}$.

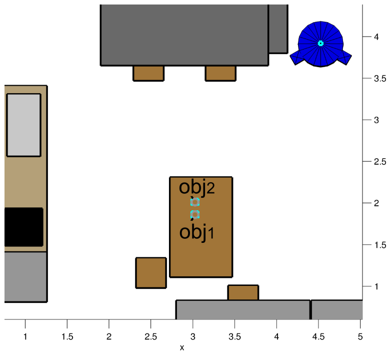


FIGURE 5.13 Scenario where two objects are located close to each other and can be grasped from a single base position. The cups are positioned at $obj_1 = \langle 3.0\text{m}, 1.85\text{m}, \frac{4}{3}\pi \rangle$ and $obj_2 = \langle 3.0\text{m}, 2.0\text{m}, \frac{2}{3}\pi \rangle$.

The cups are close to each other and the handles are oriented in a way that it seems possible to grasp both objects from a single base position. In order to find such base positions, the robot has to compute ARPLACEs for grasping each object individually, and then search for grid cells that have a grasp success probability of $> 0\%$ in both ARPLACEs. The ARPLACE probability distribution for grasping obj_1 is depicted in the left plot of Figure 5.14, and the ARPLACE probability distribution for grasping obj_2 is depicted in the right plot of Figure 5.14. It can be seen that obj_1 can be grasped with the right arm from table edge 1 and table edge 4, and with the left arm from table edge 3. obj_2 can be grasped with the right arm from table edge 3 and table edge 4, and with the left arm from table edge 1.

The next step is to compute grasp success probability for grasping both objects from the same grid cell. The computed grasp success probability should reflect the probability that *all grasps succeed*. Therefore, the grasp success probabilities of grasping individual objects are multiplied. Applied to the problem of grasping two objects at once this means that grasp success probability of a grid cell (x, y) for successfully grasping both objects at once $(p_{obj_{12}}(x, y))$

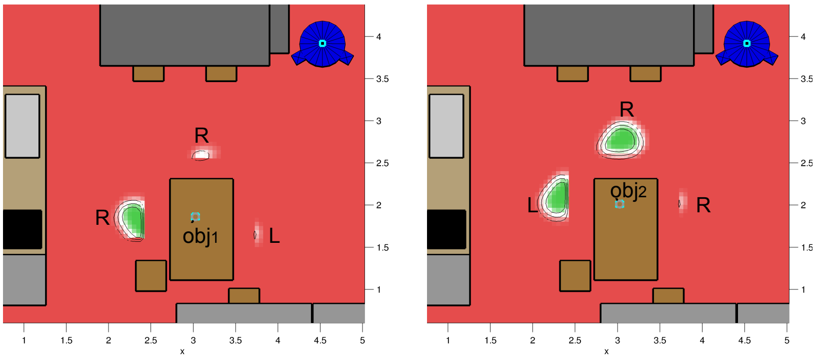


FIGURE 5.14 ARPLACEs for individually grasping the target objects. Left: ARPLACE for grasping obj_1. Right: ARPLACE for grasping obj_2. The letters *R* and *L* next to ARPLACE clusters indicate if the cluster is for grasping the object with the *R*ight or *L*eft arm.

is the product of grasp success probability for obj_1 ($p_{obj_1}(x, y)$) and grasp success probability for obj_2 ($p_{obj_2}(x, y)$). A naive approach to compute $p_{obj_{12}}(x, y)$ would be

$$p_{obj_{12}}(x, y) = p_{obj_1}(x, y) \cdot p_{obj_2}(x, y)$$

However, there is a constraint to that definition, because it is not possible to grasp two objects with the same arm. There are only two valid schemes for grasping both objects at once.

- Grasp obj_1 with the right arm and obj_2 with the left arm (abbreviated with *RL*).
- Grasp obj_1 with the left arm and obj_2 with the right arm (abbreviated with *LR*).

The abbreviations *RL* and *LR* specify the order of how to grasp the objects. The first letter refers to the arm of grasping obj_1, and the second letter refers to the arm of grasping obj_2. The correct formula for computing grasp success probability for grasping two objects from a single manipulation place is as follows

$$p_{obj_{12}}(x, y) = max(p^{s_1}_{obj_1}(x, y) \cdot p^{s_2}_{obj_2}(x, y)) \; ; \; s_1, s_2 \in \{L, R\}; s_1 \neq s_2 \tag{5.4}$$

where $p_{obj_{12}}(x, y)$ is grasp success probability of grasping both objects from grid cell (x, y), s_1 and s_2 specify the arm side for grasping, $p^{s_1}_{obj_1}(x, y)$ is grasp success probability of grasping obj_1 with any arm from grid cell (x, y), and $p^{s_1}_{obj_2}(x, y)$ is grasp success probability of grasping obj_2 with any arm from grid cell (x, y). The constraint "$s_1, s_2 \in \{L, R\}; s_1 \neq s_2$" prevents that multiple objects are grasped with the same manipulator. The maximum operator is required because in case the target objects can be grasped with both arm combinations (RL and LR), then the more promising combination is preferred.

We illustrate equation 5.4 by presenting an example. Figure 5.15 depicts the computations for grasping obj_1 with the right arm and obj_2 with the left arm.

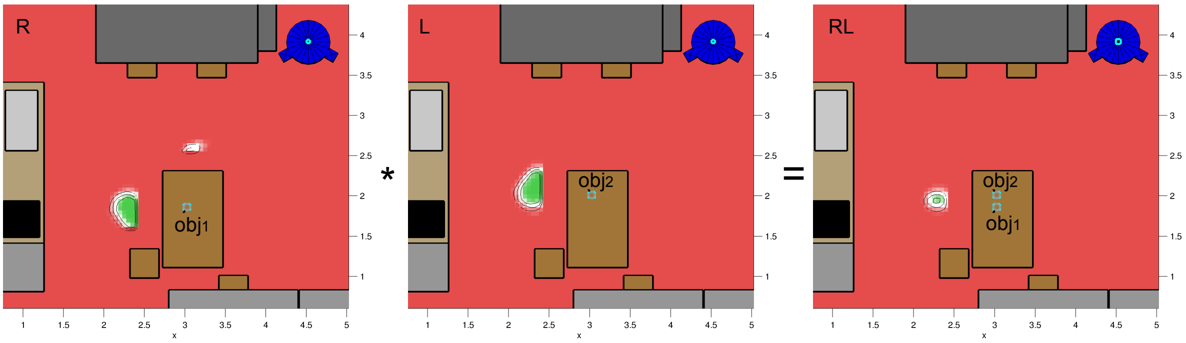


FIGURE 5.15 Computing ARPlaces for grasping obj_1 with the right arm and obj_2 with the left arm (grasp scheme *RL*). Symbols in the top left corner indicate which arm is used for grasping. Left: ARPlace for grasping obj_1 with the right arm. Center: ARPlace for grasping obj_2 with the left arm. Right: ARPlace for grasping obj_1 with the right arm and obj_2 with the left arm from a single base position. This is the combination of the left and center plot through equation 5.4.

It can be seen that both target objects can be grasped at once, if the robot performs the manipulation action from table edge 1. There are even several grid cells that predict a grasp success probability of more than 80%. The reason why multi-arm manipulation is possible is that the clusters with grasp success probabilities of more than 0% of the left and center plot in Figure 5.15 overlap, and therefore the product of these grasp success probabilities is also above 0%. However, this is only the case for the ARPlace clusters at table edge 1. The ARPlace cluster for grasping obj_1 with the right arm from table edge 4 is cancelled out because obj_2 cannot be grasped with the left arm from table edge 4.

Figure 5.16 depicts the ARPlace probability distributions for grasping obj_1 with the left arm (left plot) and obj_2 with the right arm (center plot). It can be seen that this grasp scheme is not possible, because the right plot of Figure 5.16 does not contain a grid cell with grasp success probability of above 0%. The reason is that there are no corresponding grid cells with grasp success probability of more than 0% in the left and center plot. There is almost an overlap at table edge 3, but a gap of 10cm remains.

Figure 5.17 depicts an example of an invalid ARPlace. The plot shows that there would be grid cells for grasping both objects with the right arm. However, obj_1 would fall to the ground when the robot opens the right gripper in order to grasp obj_2.

When computing the ARPlace for grasping both objects at once, we merge the ARPlaces for the valid grasping schemes *RL* and *LR*. We do this by using the max operator. For every grid cell, the higher grasp success probability is stored together with meta information that

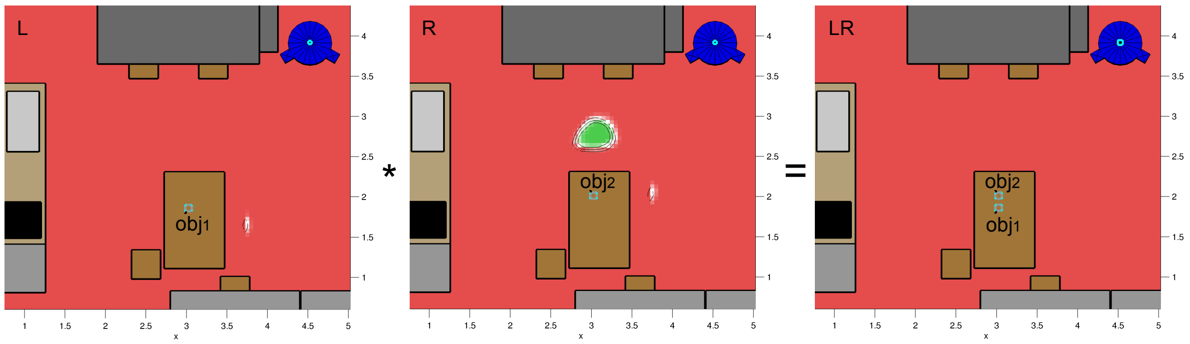


Figure 5.16 Left: ARPlace for grasping obj_1 with the left arm. Center: ARPlace for grasping obj_2 with the right arm. Right: ARPlace for grasping both objects at once.

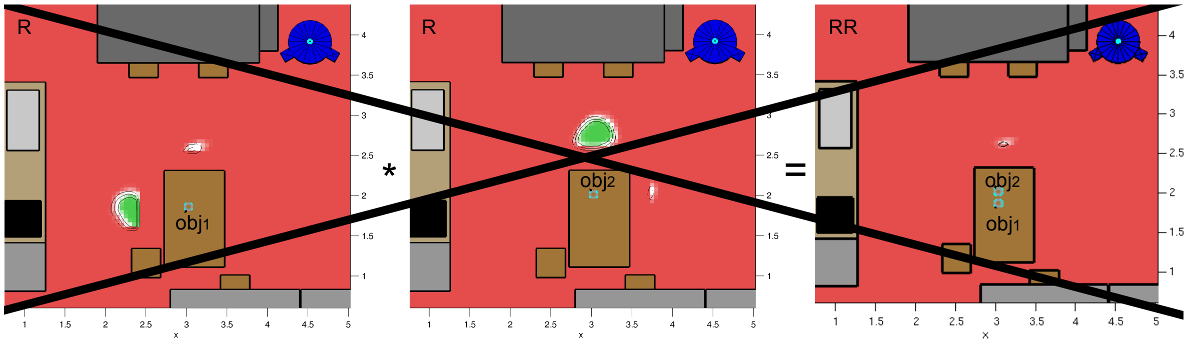


Figure 5.17 Left: ARPlace for grasping obj_1 with the right arm. Center: ARPlace for grasping obj_2 with the right arm. Right: Invalid ARPlace for grasping both objects at once.

indicates if the corresponding grasp success probability was derived from the *RL* or *LR* grasping scheme. In this example, the merged ARPlace is identical to the ARPlace for the RL grasping scheme, because the ARPlace for the LR grasping scheme is zero everywhere.

Figure 5.18 depicts a more interesting example. The task is to grasp two glasses by approaching them from the top. The left and center plot show that both objects can be grasped either with the *RL* or *LR* grasp scheme. At table edge 1, grasp success probability for the *LR* grasp scheme is *included* in grasp success probability for the *RL* grasp scheme, meaning that

- the set of promising grid cells for the *LR* grasping scheme is a subset of the set of promising grid cells for the *RL* grasping scheme
- grasp success probability of every grid cell (x, y) is higher for the *RL* grasp scheme than for the *LR* grasp scheme

That is why in the resulting ARPlace probability distribution that is plotted in the right plot of Figure 5.18, the ARPlace probability distribution at table edge 1 looks identical to that of the *RL* grasping scheme. It is labeled accordingly. For grasping from table edge 3 it is the other way around. Grasp success probability for the *RL* grasping scheme is included in the *LR* grasping scheme. Therefore, the resulting ARPLACE probability distribution at table edge 3 looks identical to that of the *LR* grasp scheme. At table edge 4, no grasping scheme is included in the other. There are grid cells with a grasp success probability of above 0% for the *RL* grasp scheme that do have a grasp success probability of 0% for the *LR* grasp scheme, and vice versa. Therefore, the resulting ARPLACE cluster at table edge 4 consists of grasp success probability values of both grasp schemes.

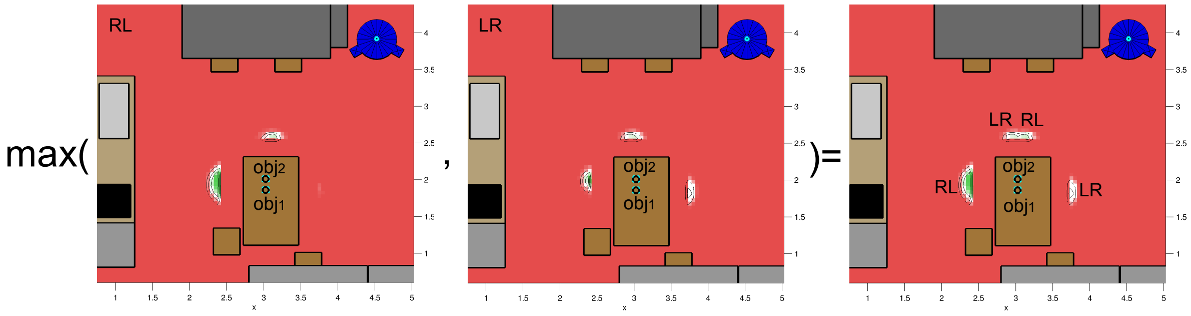


FIGURE 5.18 Left: ARPLACE for grasping both objects with the RL grasping scheme. Center: ARPLACE for grasping both objects with the LR grasping scheme. Right: ARPLACE for grasping both objects at once. The labels *RL* and *LR* represent meta information that is stored for every grid cell in order to know if the corresponding grasp success probability refers to the *RL* or *LR* grasping scheme.

5.6.2 Evaluation

In this section we evaluate the impact of different object configurations on ARPLACEs for multi-arm manipulation. We examine (1) the impact of different distances between two objects; (2) the impact of the distance of two objects to the table edge, and (3) the impact of different angular orientations of two objects. We use cups as objects.

5.6.2.1 Impact of Object Distance

Figure 5.19 shows several ARPLACEs for grasping two cups at once. The first cup is always positioned at $obj_1 = \langle 3.0\text{m}, 1.8\text{m}, \frac{4}{3}\pi\rangle$ with respect to the world frame, while the pose of the second cup changes from $obj_2 = \langle 3.0\text{m}, 1.9\text{m}, \frac{2}{3}\pi\rangle$ to $obj_2 = \langle 3.0\text{m}, 2.15\text{m}, \frac{2}{3}\pi\rangle$ with increments of 5cm along the y-axis. Therefore, the distance between both cups varies from 0.1m to 0.35m.

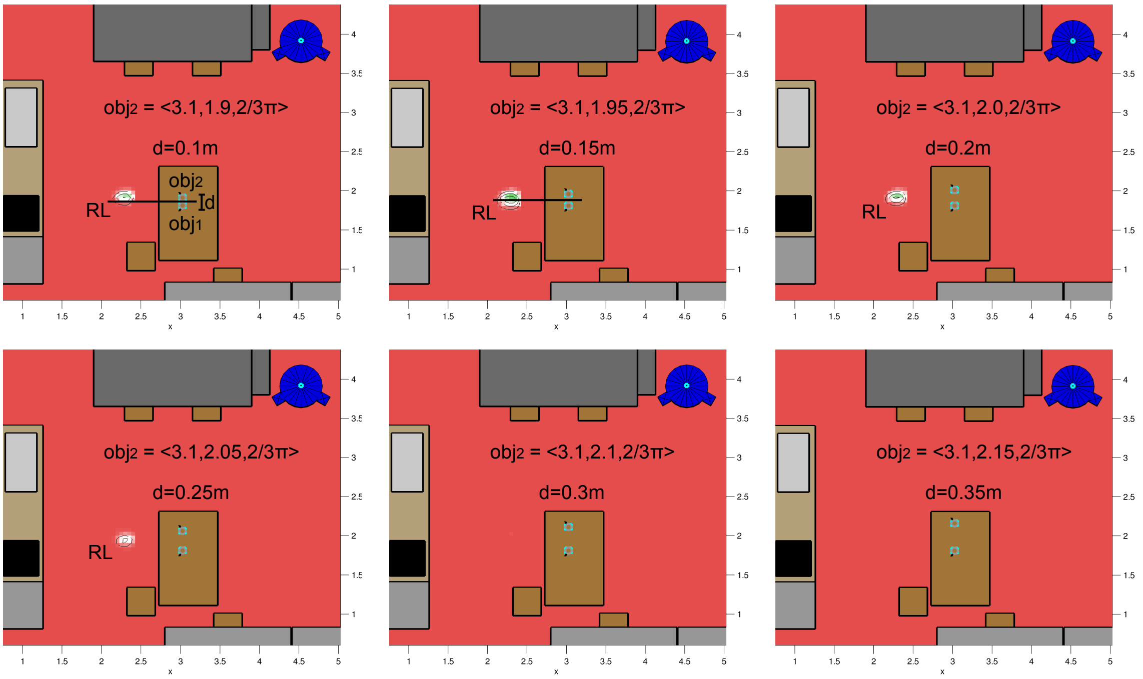


FIGURE 5.19 Merged ARPLACEs for different cup configurations. The pose of the first cup is always $obj_1 = \langle 3.0\text{m}, 1.8\text{m}, \frac{4}{3}\pi \rangle$. The pose of the second cup (obj_2) and the distance between the cups (d) is denoted in the images. The black line in the first two plots visualizes that ARPLACE proposes the robot to navigate to base positions that are approximately between the cups.

It can be seen that the most promising grid cells in all ARPLACEs are approximately in between both cups (visualized by a black line in the first two plots). The ARPLACE proposes the robot to navigate to base positions where the distance of the arms to both cups is approximately equal. All ARPLACEs prefer the RL grasp scheme.

The ARPLACE for a cup distance of 0.15m has the largest area of promising manipulation places and reaches the highest maximal grasp success probability. The ARPLACEs for a cup distance of 0.1m and 0.2m have approximately equal area of promising manipulation places, with a slight advantage in maximal grasp success probability for the ARPLACE with a cup distance of 0.2m. The ARPLACE for a cup distance of 0.25m is the first that does not reach a maximal grasp success probability of 80%. When the cup's distance reaches or exceeds 0.3m, ARPLACE does not find promising manipulation places for grasping both objects at once any more.

Overall, the ARPLACE predicts that the robot will be successful if the cups are close to each other. Grasp success probability is predicted to decrease, as cup distance increases. The maximal cup distance where ARPLACE assumes the robot being able to successfully grasp both cups at once is between 0.25m and 0.3m.

5.6.2.2 Impact of Distance to Table Edge

The next criterion we want to evaluate is the distance of the objects to the table edge, as depicted in Figure 5.20. The distance of the cups to the table edge $\Delta x_{obj_1} = \Delta x_{obj_2} = d$ is set to $d = 0.3725\text{m}$, $d = 0.2725\text{m}$, and $d = 0.1725\text{m}$, while the cup's distance to each other and the angular orientations of the handles remain the same.

It can be seen that the ARPLACE probability distribution for $d = 0.1725\text{m}$ and $d = 0.2725\text{m}$ are nearly identical. A difference is that the ARPLACE for $d = 0.2725\text{m}$ is shifted approximately 10cm towards the table, in order to compensate for the increased distance of the objects to the table edge. When the distance to the table edge increases further to $d = 0.3725\text{m}$, then parts at the front of the ARPLACE probability distribution are cut off because the robot would bump into the table, and the overall area of the ARPLACE therefore shrinks. The most promising grid cells are again in between both cups for all three ARPLACEs.

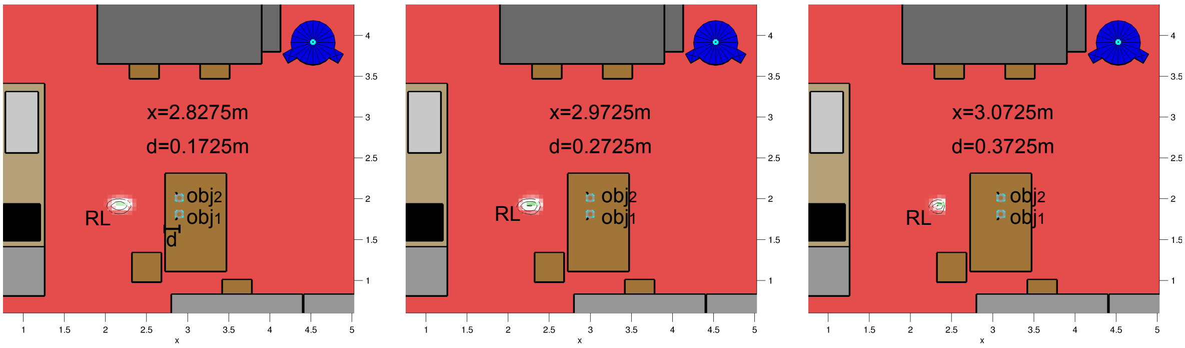


FIGURE 5.20 Merged ARPLACEs for different distances of the cups to the table edge. The poses of the cups are $obj_1 = \langle x, 1.8\text{m}, \frac{4}{3}\pi \rangle$, and $obj_2 = \langle x, 2.0\text{m}, \frac{2}{3}\pi \rangle$ with the x-coordinate being set to 2.8725m, 2.9725m, and 3.0725m with respect to the world frame. The x-value and the distance of the objects to the table edge d is specified in each plot.

5.6.2.3 Further Scenarios

Figure 5.21 depicts three further scenarios. The objects are positioned at varying poses. In the first scenario the cup orientations are varied. When comparing the new orientation to earlier cup poses in this evaluation, both cups are turned clockwise. Therefore, the robot has to grasp obj_2 more from the back, requiring the robot to grasp around obj_2 and having to perform the manipulation action more closely to obj_2. obj_1 on the other side has to be approached more from the front than before. As a result, the merged ARPLACE is not between the cups any more, but shifted towards obj_2 as indicated by the black line in the left plot of Figure 5.21.

In the second scenario, the cup handles were rotated so that each handle points towards the other cup, and not away from it. The result is that the merged ARPLACE proposes to grasp

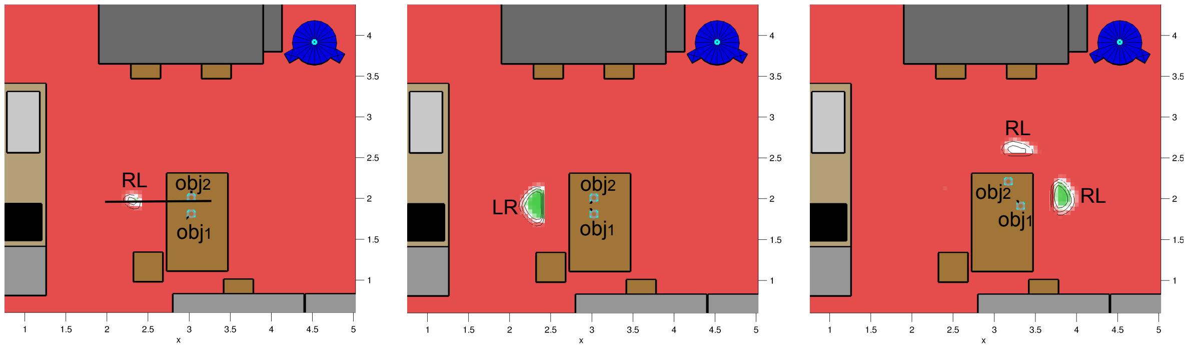


FIGURE 5.21 Merged ARPLACEs for different object poses. Left: 1st scenario. $obj_1 = \langle 3.0\mathrm{m}, 1.8\mathrm{m}, \frac{5}{4}\pi\rangle$, and $obj_2 = \langle 3.0\mathrm{m}, 2.0\mathrm{m}, \frac{1}{2}\pi\rangle$. The black line depicts that the ARPLACE is shifted towards obj_2. Center: 2nd scenario. $obj_1 = \langle 3.0\mathrm{m}, 1.8\mathrm{m}, \frac{2}{3}\pi\rangle$ and $obj_2 = \langle 3.0\mathrm{m}, 2.0\mathrm{m}, \frac{4}{3}\pi\rangle$. Right: 3rd scenario. $obj_1 = \langle 3.3\mathrm{m}, 1.9\mathrm{m}, \frac{2}{3}\pi\rangle$ and $obj_2 = \langle 3.15\mathrm{m}, 2.2\mathrm{m}, \frac{5}{3}\pi\rangle$.

obj_1 with the left arm and obj_2 with the right arm. Because the orientation of the cups' handles is symmetric, the ARPLACE proposes to position the robot's base between the cups. Please note that the LR grasping scheme will probably not enable the robot to perform the grasping actions in parallel, but to perform them sequentially in order to prevent self collision of the manipulators.

In the third scenario the robot has to grasp a cup (obj_1) and a glass (obj_2). ARPLACE clusters emerge at table edge 3 and 4 and the overall area of promising manipulation places increases when compared to the scenarios of grasping two cups. The reason is that a glass can be grasped from the top which puts fewer kinematic constraints on the manipulator.

To conclude this section, we evaluated scenarios where two target objects were positioned at various configurations. Multiple target objects can only be grasped at once when they are near each other. If the distance between objects exceeds a certain threshold, multi-arm grasping is not possible anymore. In the scenario presented in section 5.6.2.1, the maximal distance for multi-arm manipulation was between 0.25m and 0.3m. In case the objects can be approached symmetrically, manipulation places between target objects are preferred, but as the objects' orientations and approach vectors change, this is not the case any more. Because grasping from the top puts fewer kinematic constraints on the manipulator, the area of promising grid cells is bigger than for grasping from the side. As a result, it is easier to grasp multiple objects when they can be approached from the top. The best base position for grasping multiple objects at once are not necessarily between the target objects. In order to decide whether it is preferable to grasp multiple objects at once or not we use a high level planning system. The high level planning system itself was implemented by other researchers of our group. That is why section 5.6.3 should not be considered as original research of this thesis, but is presented to give a comprehensive overview of the ARPLACE framework. For detailed information about the high level planning system, please refer to the work of Mösenlechner and Beetz (2009) and Müller (2008).

5.6.3 High Level Planning System

In the last section we computed ARPLACEs for grasping multiple objects at once. We found that grasp success probability for multi-arm manipulation is computed as the product of grasp success probabilities for grasping single objects. Please note that multiplying two input probability values always leads to an output probability that is equal or less than either input probability. Therefore grasp success probability for grasping multiple objects at once can not be higher than grasping either object separately. This is an intuitive but important result, because it never makes sense to grasp multiple objects at once when the only goal is to maximize grasp success probability. However, we described that multi-arm manipulation can significantly reduce the amount of execution time. The challenge is how the robot decides whether to prefer sequential manipulation to maximize grasp success probability or parallel manipulation to minimize execution time.

In this section we present an approach where this decision is made by a transformational planning system. The transformational planning system requests the ARPLACE framework to compute promising base positions for a given manipulation task. The ARPLACE framework then computes ARPLACE probability distributions for grasping objects sequentially, and probability distributions for grasping multiple objects at once and returns them. Finally, the transformational planning system analyzes the results and chooses the most suitable base position.

We explain how plans are represented in section 5.6.3.1, how plans are modified in general by a transformational planner in section 5.6.3.2, and how the transformational planner specifically handles ARPLACE probability distributions for finding optimal manipulation places in section 5.6.3.3.

5.6.3.1 Plan Design

We define plans as robot control programs that cannot only be executed, but also reasoned about. This is important, because it enables a transformational planner to reason about the intention of a specific code part. Standard control programs written in the Lisp dialect RPL, that was developed by McDermott (1991), are annotated in order to indicate their purpose and make them transparent to the transformational planner. For example, manipulation actions that must be performed from a certain base position are executed within the context of an *at-location* block. The most important RPL instructions for semantic annotation in the context of mobile manipulation are *achieve*, *perceive* and *at-location*.

The *achieve* statement asserts that the logical expression within the *achieve* statement holds after execution. This means that we can test if a certain action like a particular manipulation action was performed successfully. For example, the statement *(achieve (entity-picked-up ?object))* states that after executing this instruction, the object referenced by variable *?object* must be in the robot's gripper. Before manipulating objects, the robot must find the objects and instantiate them in its belief state. The statement *(achieve (perceive ?object))* guarantees that after executing it, the object referenced by *?object* has been found and a reference to its internal representation is returned.

Mobile manipulation implies the execution of actions from specific locations. Therefore, it must be guaranteed that grasping actions are only executed when the robot is at a specific location. *(at-location ?location ...)* asserts that code within its context is either executed at the specified location, or fails. Please note that plan transformations which change the location of actions, directly modify the *?location* parameter of *at-location* expressions. Therefore, *at-location* is the most important declarative plan expression for optimizing ARPLACEs.

5.6.3.2 Transformational Planning System

Transformational planning enables robots to detect and fix behavior flaws, such as 1) collisions that can be caused by under-specified goal locations; 2) blocked goals, like when a chair is positioned at a location the robot wants to navigate to; 3) flaws affecting performance. Our task of deciding, whether to sequentially or parallely grasp objects is a member of the third category, because a plan that decides to sequentially grasp objects will take longer to execute than a plan that saves a lot of navigation actions and enables parallel grasping. Additional execution time is unnecessary if grasp success probability for multi-arm manipulation is not much worse, and should be considered as a performance flaw.

A transformational planning system consists of three main components

A) a projection mechanism for predicting the outcome of a plan

B) a mechanism for detecting flaws within the predicted plan outcome

C) mechanisms to fix detected flaws by applying transformation rules to the plan code

Planning is performed by repeatedly performing these steps until the resulting plan cannot be further optimized. Subsequently the plan is executed. A visualization of this process is depicted in Figure 5.22.

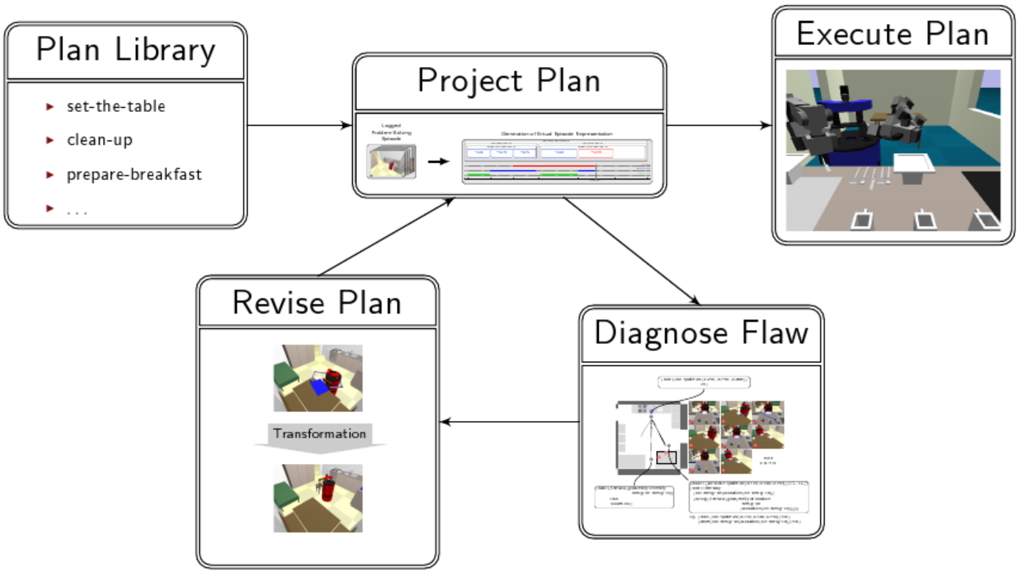


FIGURE 5.22 Overview of how the transformational planning system optimizes plans.

ad A) Plan Projection: One component of a transformational planning system is an accurate prediction mechanism that generates a temporally ordered set of events. For projecting plans, we use the Gazebo simulator. For every point in time the projection of a plan generates an execution trace that contains the state of the plan, the robot's belief state, and the state of the simulated world.

ad B) Behavior Flaws and Reasoning about Plan Execution: The second component of a transformational planner is a reasoning engine that finds pre-defined flaws in robot behavior. As mentioned earlier, we are interested in fixing performance flaws. Listing 5.1 shows the specification of the performance flaw that is created when two manipulation actions are executed at different initial locations.

```
(def-behavior-flaw unoptimized-locations
  :specializes performance-flaw
  :flaw (and
          (task-goal ?task-1
            (achieve (entity-picked-up ?object-1)))
          (task-goal ?task-2
            (achieve (entity-picked-up ?object-2)))
          (thnot (== ?task-1 ?task-2))
          (optimized-action-location
            ?object-1 ?object-2
            ?optimized-location)))
```

LISTING 5.1 Definition of a performance flaw.

More specifically, listing 5.1 shows the definition of behavior flaws. The flaw is in the class of performance flaws, specializing the flaw *performance-flaw* (line 2). In lines 4 to 8, two different grasping tasks are matched, and the corresponding variables are bound. In line 9, the ARPLACE framework is queried for a manipulation place for grasping both objects at once, *?object-1* and *?object-2*. The predicate only holds true when the probability of this manipulation place is sufficiently high (>0.85). This means that grasping two objects sequentially will not be transformed, when there is no good place for multi-arm manipulation. Note that we used a hard-coded parameter for defining, what is "sufficiently high". However, the real value for "sufficiently high" depends on the task context. In robotic soccer for instance, it can be beneficial to choose fast and risky moves, whereas in human-robot interaction certainty of successful execution is more important than speed.

ad C) Plan Transformations and Transformation Rules: After a flaw has been detected, the planner applies a transformation rule to the plan code in order to fix the flaw. A transformation rule consists of an input schema, a transformation part, and an output plan.

$$\frac{\text{Input Schema}}{\text{Output Plan}}\Big]\,\text{Transformation}$$

The input schema is matched against the plan part with the flaw. If they match the transformation part is applied to the input schema. The result is new plan code, which is the output plan that achieves the same goals as the input scheme but got rid of flaws. Listing 5.2 shows the transformation rule for fixing the flaw shown in listing 5.1.

```
(def-tr-rule fix-unoptimized-locations
   :input-schema
      ((and (task-goal ?location-task-1
                   (at-location (?location-1) . ?code-1))
                (sub-task ?location-task-1 ?task-1))
        (and (task-goal ?location-task-2
                  (at-location (?location-2) . ?code-2))
                (sub-task ?location-task-2 ?task-2)))
  :output-plan
     ((at-location (?optimized-location) . ?code-1)
      (at-location (?optimized-location) . ?code-2)))
```

LISTING 5.2 Transformation rule for fixing the performance flaw of performing manipulation actions from different base positions.

Applied to the task of merging ARPLACEs, the transformation rule would expand to the plan transformation in Figure 5.23.

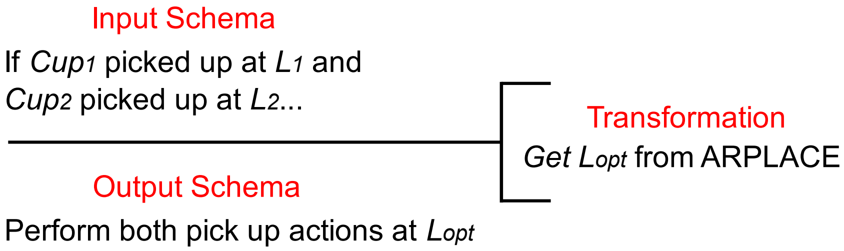


FIGURE 5.23 Plan transformation for performing multi-arm manipulation.

5.6.3.3 Optimizing ARPLACES

ARPLACES are not only integrated into the robot control program, they are also integrated into the reasoning engine of the transformational planner. Using two manipulation places for grasping is considered a performance flaw if both grasps could be performed from a single base position. A parser investigates the execution trace for the occurrence of two different grasping actions, where one is executed at location L_1, and the other is executed at location L_2. Then we query ARPLACE for a location L_3 which is the manipulation place with the highest grasp success probability for performing both actions at once. If grasp success probability is sufficiently high, we apply a plan transformation and replace locations L_1 and L_2 by location L_3.

We evaluated the merging of ARPLACES for grasping two objects at once. Two cups were placed on a table, and the distance between them was varied between 20cm and 60cm, in increments of 5cm. Average execution time of the following action sequences was evaluated

- Move to best manipulation place of object1; Grasp object1; Move to best manipulation place of object2; Grasp object2
- Move to best manipulation place between object1 and object2; Grasp object1 and object2 in parallel

Our evaluation shows that grasping two cups from different base positions requires 48 seconds on average. The variance is low because the grasping actions take almost equal times. The parameter that varies is the distance between the cups, but the second navigation action takes almost equal time because the effect of moving 20cm or 60cm is negligible. When we applied transformation rules for performing multi-arm manipulation, average execution time was reduced to 32 seconds, which is a significant performance gain (t-test: $p < 0.001$). Please note that the transformational planner never tried to grasp both cups at once, when the distance between cups was above 45cm. The reason is that no grid cell with sufficiently high grasp success probability existed.

5.7 Dealing with Uncertainty into the Object's Type

Many everyday activities do not specify which objects are involved. A task like "clean the table" for example, does not tell the robot which objects it will have to manipulate. Imagine a robot moving through a kitchen door in order to clean a table. The robot detects an object that is located on a table. Because the robot is far away its vision system is not able to fully recognize the type of the object. It rather predicts that the target object is a glass with a probability of 70% or a cup with a probability of 30%. The problem with this predicition is that the grasp point and approach vector for these objects are different. The robot would prefer to grasp a cup at its handle by approaching it from the side. A glass is better grasped by approaching it from the top. Because there are different Generalized Success Models to handle grasps from the side and from the top, the robot is uncertain which ARPLACE to compute. A possible solution would be to compute the ARPLACE for grasping the object with higher probability. In this example this would mean to compute an ARPLACE for grasping a glass. But this would also mean to throw away information. A superior approach is to compute both ARPLACE probability distributions and merge them as the sum that is weighted by the corresponding object probabilities.

In general, if a robot estimates the type of a target object among n objects (o_1,..,o_n) that are known to the robot, then the robot ends up with an object probability vector $(p_1, .., p_n)$. The corresponding ARPLACE probability distributions $A_1, .., A_n$ are then merged by computing each grid cell's grasp success probability $p_{(x,y)}$ as follows

$$\forall(x, y) : p_{(x,y)} = \sum_{i=1}^{n} p_i \cdot A^i_{(x,y)} \tag{5.5}$$

where $A^i_{(x,y)}$ is the grasp success probability at grid cell (x, y) in the ARPLACE computed for object o_i.

Figure 5.24 depicts the process of merging ARPLACEs if the robot is uncertain about the target object's type. The left image depicts the ARPLACE probability distribution for the assumption that the target object is a cup. This would require the robot to grasp the object from the side. The center image shows the ARPLACE probability distribution for the assumption that the target object is a glass, which leads to the result that the robot grasps it from the top. The right image shows the resulting ARPLACE probability distribution. It can be seen that the grid cells with the highest grasp success probability are in a region, where the ARPLACEs for side and top grasps overlap. This is intuitive, because if there are places which are promising for grasping any kind of object, then the robot should prefer to go there.

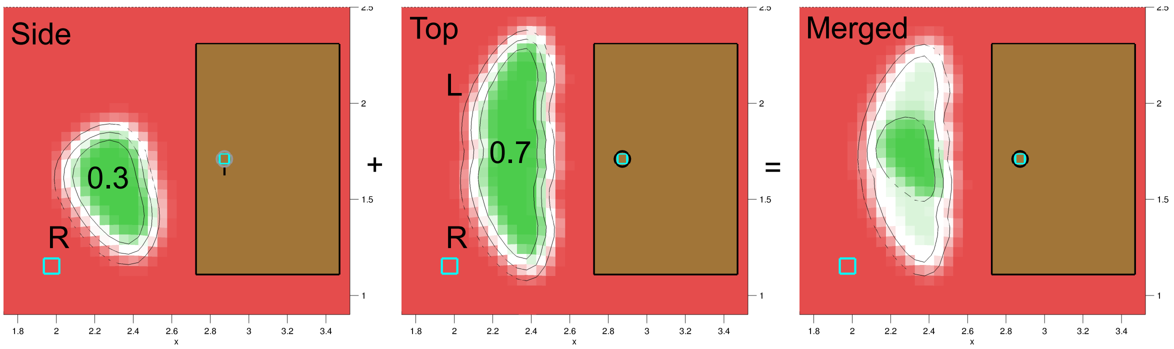


Figure 5.24 Merging ARPlaces when the robot is uncertain about the target object's type. Left image: ARPlace for grasping a cup from the side. Center: ARPlace for grasping a glass from the top. Right: ARPlace when the robot estimates the target object to be a cup with 30% probability and a glass with 70% probability. The label next to the ARPlace cluster indicates the arm side for grasping. The number inside the ARPlace cluster represents the probability that the robot attributes to the object type.

5.8 Integration of Unexpected Experience

In section 4.4.2 we discussed that a manipulation action may fail unexpectedly because the robot made errors in the state estimation process or effects occured that are not familiar to the robot. The problem with unexpected failure is that the high-level planning system does not know why the failure happened and has to guess about the reasons. Maybe learning a new Generalized Success Model helps, but this requires a significant amount of time and should be delayed to idle periods.

In order to enable the robot to act immediately, a repeller probability distribution is introduced as an ad-hoc solution. The reason why a repellor distribution is required is that when a manipulation action failed and ARPlace is immediately queried for another promising base position to re-try grasping the object, then ARPlace will return the same or – due to sampling variances – very similar manipulation places. The repeller probability distribution is an ad-hoc approach that pushes the robot away from the current grid cell where the manipulation action failed.

The left plot in Figure 5.25 depicts an ARPlace probability distribution. It can be seen that three ARPlace clusters emerge, because the cup is graspable from table edge 1, table edge 3, and table edge 4. The robot located the target object correctly, but overlooked the chair at the table. That is why the ARPlace cluster at table edge 1 has a larger area and bigger maximal grasp success probability than it should reasonably have. The ARPlace cluster also includes the grid cell with maximal grasp success probability. Therefore, the robot tries to grasp the cup from table edge 1 and bumps into the chair.

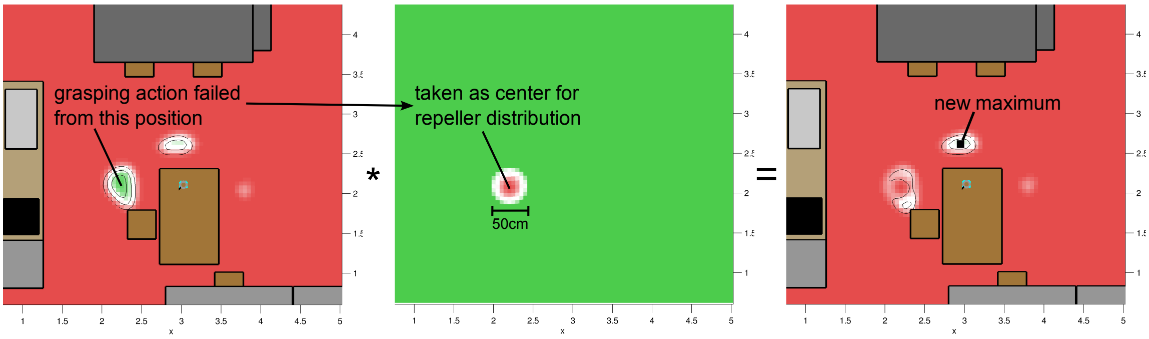


FIGURE 5.25 Left: Original ARPLACE probability distribution. Center: Repeller probability distribution with the priorly proposed base position as origin and a diameter of 50cm. Right: ARPLACE that considers the repeller probability distribution. It is the result of multiplying the probabilities of the left and center plot.

Because the task is urgent, the robot wants to re-try the manipulation action immdiately but tells the ARPLACE framework that the manipulation action failed from the priorly proposed base position. The ARPLACE framework therefore computes a repeller probability distribution (p_R), as seen from a top-down view in the center plot of Figure 5.25 and from an isometric perspective in the left plot of Figure 5.26. The repeller probability distribution is implemented as a paraboloid of revolution with the priorly proposed base position defining the origin. This origin has a probability value of 0.0. A parameter specifies the radius until the paraboloid of revolution reaches a probability value of 1.0. Probability values above 1.0 are mapped to 1.0.

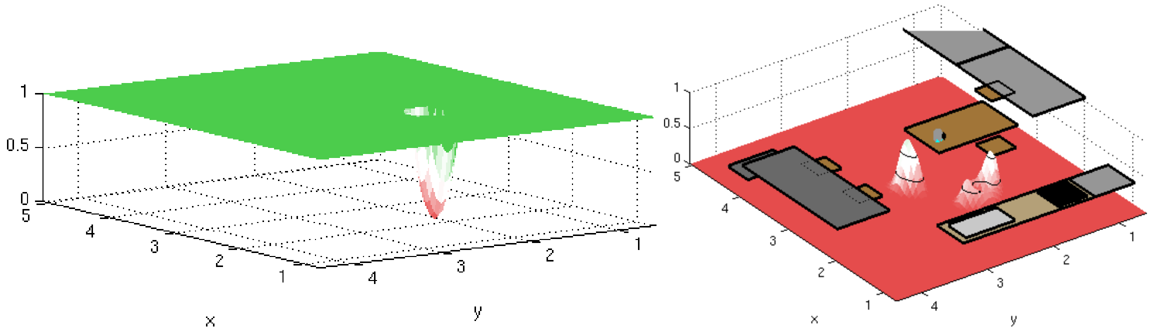


FIGURE 5.26 Left: Isometric version of repeller probability distribution shown in center plot of Figure 5.25. Right: Isometric version of grasp success probability shown in right plot of Figure 5.25.

In order to compute an ARPLACE probability distribution p' that considers the repeller probability distribution, the probability value of every grid cell in the original ARPLACE probability distribution $p(x, y)$ is multiplied with the probability value of the corresponding grid cell in the repeller probability distribution $p_R(x, y)$.

$$\forall(x, y) : p'(x, y) = p(x, y) \cdot p_R(x, y) \tag{5.6}$$

The new ARPLACE probability distribution is shown from a top-down view in the right plot of Figure 5.25 and from an isometric perspective in the right plot of Figure 5.26. It can be seen that the repeller probability distribution knocked out the region with maximal grasp success probability. The new grid cell with maximal grasp success probability is now found in the ARPLACE cluster at table edge 4.

Repeller probability distributions do not significantly deteriorate computation time of Action-Related Places. For an overall of 25 experiments, the mean computation time was 0.42ms with a standard deviation of 0.06ms.

CHAPTER 6

Utility Framework for Action-Related Places

In the last two chapter chapters, we showed how to compute and refine ARPLACEs that are based on grasp success probability. We mapped grid cells to the predicted probability that a manipulation action will succeed when it is performed from this grid cell (see Figure 6.1). By optimizing grasp success probability more robust mobile manipulation is achieved.

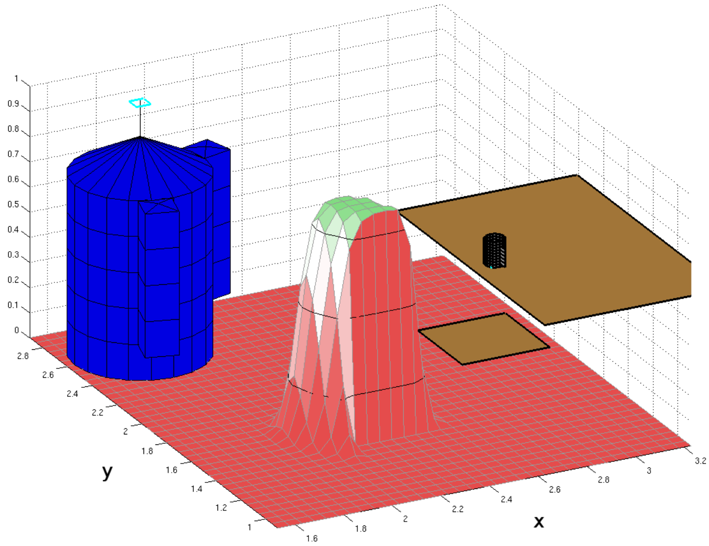


FIGURE 6.1 The robot, a chair, a table, and a ARPLACE based on grasp success probability for grasping the cup on the table. The ARPLACE discretizes space into grid cells, where each cell represents the predicted probability of successfully grasping the cup when the grasping action is performed from a base position within this cell. A hole is bumped into the distribution of grasp success, because the chair blocks several promising grid cells.

However, grasp success may not be a robot's only concern. Sisbot demonstrated that in the presence of humans, the robot should prefer to stay within the humans' field of view, especially when moving around (Sisbot et al., 2007) or interacting with humans (Sisbot, 2008). If a task is urgent, performing the task as quickly as possible has priority. On the other hand, if the robot's battery is low, saving energy becomes a vital goal. But how can we evaluate whether it is preferable for a robot to perform a manipulation action quickly but with the drawback of higher energy consumption, or slowly but with accordingly reduced energy consumption? We do so by using *decision theory* that was introduced by von Neumann and Morgenstern (1944). The core principle of decision theory is to assign *comparable* utilities to each of the potentially conflicting task goals and to subsequently maximize expected utility.

In this chapter, we extend and generalize the concept of probabilistic ARPLACEs to a utility framework. Utility-based ARPLACEs are thus able to represent arbitrary task constraints, which allows robots to make the most sensible trade-off between them by computing overall expected utility. The utilities we consider here are travel time and utility of successful grasping. Taking these utilites into account enables the robot to trade off efficiency and robustness during planning and navigation. Utility-based ARPLACEs therefore apply to a much broader range of tasks and goals. Figure 6.2 depicts the computational process for computing utility-based ARPLACEs and serves as an outline of this chapter.

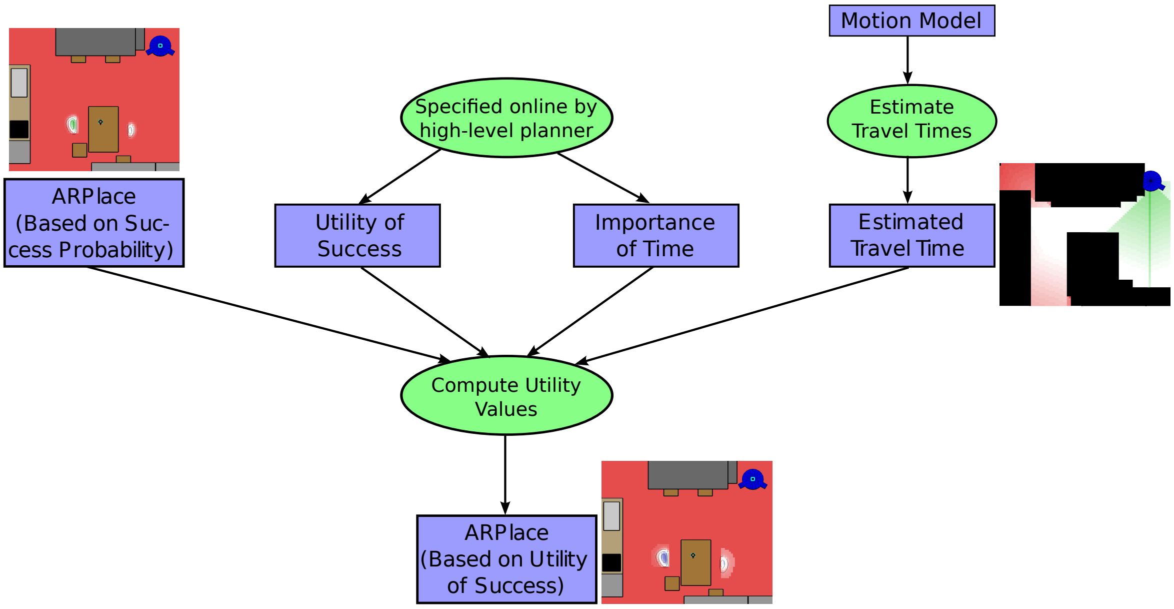

Motion Model
Specified online by high-level planner
Estimate Travel Times
ARPlace (Based on Success Probability)
Utility of Success
Importance of Time
Estimated Travel Time
Compute Utility Values
ARPlace (Based on Utility of Success)

FIGURE 6.2 Overview of computational process for computing utility-based ARPLACEs. Green circles represent algorithms that create and transform data. Blue rectangles represent data that is generated and passed from one algorithm to another. Images near blue rectangles are exemplary visualizations of the corresponding data structure.

The remainder of this chapter is structured as follows. In the next section, we present related work. Section 6.2 introduces the utility framework where section 6.2.1 describes how utilities pertaining to heterogeneous objectives can be unified. The computation of the utilities pertaining to success and execution time is described in sections 6.2.2 and 6.2.3. An evaluation in section 6.3 illustrates the advantages of utility-based ARPLACES.

6.1 Related Work

Maximizing expected utility of actions is the core principle of decision theory. Decision theory tackles the problem of making optimal decisions in the presence of uncertainty and multiple, potentially conflicting task goals. Decision theory has been widely studied in the field of economics by Smith (1988) and in the field of game theory by von Neumann and Morgenstern (1944). The utility framework that is introduced in this thesis is inspired by influence diagrams, as introduced by Howard and Matheson (1984).

In robotics, Dias and Stentz (2000) used decision theory for coordinating large groups of robots. Gerkey and Matarić (2003) employ decision theory in order to solve the problem of task allocation in multi-robot systems. Zlot et al. (2002) applied a market approach to the problem of multi-robot mapping and exploration. Their market architecture tries to maximize information gained while minimizing travel costs, thus aiming to maximize utility. We also consider the utility/costs of movement, but use an accurate time-based cost that enables us to measure utility of time more precisely in time-based cost scale, namely seconds. Zlot et al. (2002) use a distance-based approximation, but admit that a time-based cost scale "facilitates a more straightforward way to prioritize some types of tasks, for example if there are other mission objectives in addition to exploration." Haddawy and Hanks (1993) compare goal-oriented and decision-theoretic agents. They show how these paradigms can be merged with a utility framework for goal-oriented agents and how to rank different plans in the presence of uncertainty and deadlines.

McFarland and Spier (1997) introduce basic cycles which represent the utility of the robot's owner. The method allows judgements about the robot's use of travel time and energy consumption. While basic cycles focus on finding optimal policies to refuel a robot, our utility framework is applied to mobile manipulation. Basic cycles assume that the goal task always succeeds, while our utility framework has a probabilistic representation of task success and takes state estimation uncertainties into account. Larkin (1981) examines the role of time and energy in decision making of anmials.

Koenig and Simmons (1996a) applied utility functions to robot navigation in indoor environments. They were able to demonstrate how utility functions can be used to model given risk attitudes and soft deadlines. While Koenig and Simmons primarily consider execution duration, our utility framework is explicitly designed to take arbitrary task constraints into account. Furthermore, Koenig and Simmons use exponential utility functions to enable risk-sensitive planning. We assume a linear utility function for execution time, yet a high-level planner is able to specify the importance of every utility component in order to reflect the relevance that this component plays in the current task. Another approach that focusses on navigation planning under uncertainty is presented by Wellman et al. (1995).

Berenson et al. (2008) deal with the problem of finding optimal start and goal configurations for manipulating objects. Their approach considers multiple criteria such as grasp quality, configuration desirability, and configuration clutter in order to optimize the grasp itself within a high-dimensional motion planning context. In contrast, we consider numerous factors in order to find optimal manipulation places taking state estimation uncertainties into account. ARPLACE probability distributions enable high-level planning systems to reason about the manipulation task at hand, and qualitative aspects such as from which table side an object should be grasped are addressed.

6.2 Utility Framework

For a given robot, a target object, and an environmental context, the framework of Action-Related Places provides a mapping $f(x, y) : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}$ from grid cells to the real numbers. In previous chapters, the function $f(x, y)$ that maps grid cells to ARPLACE values was based on grasp success probability ($p(x, y)$). The proposed manipulation place was the grid cell that maximized grasp success probability. In the following, we extend the notion of Action-Related Places to a utility framework in which we compute the expected utility $u(x, y)$ of grid cells. The utility framework is schematically depicted in Figure 6.3.

Formally, we compute the utility of performing a grasping action at grid cell (x, y) as

$$u(x, y) = \begin{cases} p(x, y) \cdot w_S + \sum_i u_i(x, y) \cdot w_i & \text{if } p(x, y) > 0 \\ -\infty & \text{if } p(x, y) = 0 \end{cases} \tag{6.1}$$

The expected utility of success $u_S(x, y)$ is therefore grasp success probability $p(x, y)$ scaled with the importance of success w_S. Furthermore, we consider other factors that influence utility but are independent of success. These supplemental utilities $u_i(x, y)$ are weighted by factors w_i that indicate their importance relative to grasp success. We effectively filter *unpromising* grid cells for which grasp success probability is 0, eliminating the possibility of failure being outweighed by other utilities. In order to maximize expected utility with respect to cur-

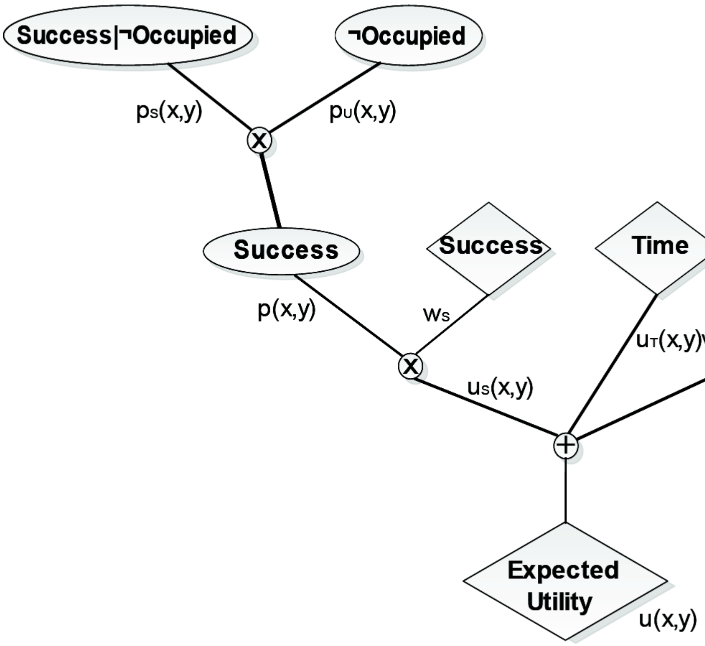


FIGURE 6.3 Overview of the utility framework, illustrating the schematic calculation of the expected utility of selecting a particular grid cell (x, y) for performing a manipulation action. Oval nodes indicate probability values while diamond-shaped nodes indicate utility values.

rent task constraints, a robot must simply find the maximal utility value u^* in the resulting ARPLACE utility distribution $u(x, y)$

$$u^* = \max_{(x,y)} u(x, y) \tag{6.2}$$

and choose the grid cell (x_{u^*}, y_{u^*}) as goal position for which the utility distribution reaches its maximum value:

$$(x_{u^*}, y_{u^*}) = \arg\max_{(x,y)} u(x, y) \tag{6.3}$$

In the following, we describe how the utility framework outlined above is implemented for a concrete robot platform in order to maximize expected utility when selecting manipulation places.

6.2.1 Unifying Heterogenous Utilities

If we are to combine the utilities pertaining to success, travel time, and energy consumption in a way that can be considered sound from a decision-theoretic point of view, then we will need to find a common unit of measurement that allows us to adequately describe any of these aspects. We found that all the aforementioned utilities can reasonably be reduced to time, and we therefore measure utility of manipulation places in seconds.

The utility pertaining to execution time is directly related to time. Because increased time should imply decreased utility and we consider linear decay of utility to be appropriate, we set the utility of time $u_T(x, y)$ to $u_T(x, y) = -t(x, y)$, where $t(x, y)$ is the time required by the robot to move to grid cell (x, y) and perform a grasp.

To define the utility of success, please observe that failing to grasp the target object implies that the grasping action will have to be repeated which requires additional time. Therefore, the utility of success can be defined as the time the robot typically saves by not having to reposition its base for a second grasp attempt plus the time for redoing the grasp itself. Let that time be $t_{\textit{redo}}$. Since a given grid cell (x, y) will lead to manipulation success only with some probability $p(x, y)$, we can expect to save only the corresponding fraction of $t_{\textit{redo}}$, which is why we set the utility to $p(x, y) \cdot t_{\textit{redo}}$ (in Equation 6.1, w_S corresponds to $t_{\textit{redo}}$).

We can define the utility of saving energy in a similar fashion: Any amount of energy $e(x, y)$ (measured in Joules) used by the robot to move to grid cell (x, y) and perform a grasp requires the robot to recharge its battery with the respective amount of energy in the future. Assuming that the robot's battery can be recharged with power P (measured in J/s), the time required to regain the energy that is lost is $e(x, y)/P$. Therefore, the utility of saving energy $u_E(x, y)$ can be computed as $u_E(x, y) = -e(x, y)/P$. As a result, the performance of battery chargers will have immediate impact on the overall utility of ARPLACE grid cells: A robot that knows that it can quickly recharge its batteries will prefer to save time, while a robot that knows that recharging is slow might trade task execution time for energy.

Once all utility components use the same unit of measurement, the sum of utility components is well-defined and the weight that we additionally assign to each of the supplemental utility components (w_i in Equation 6.1) simply becomes the importance that we attribute to the respective component relative to grasp success. By default, weights should be 1. However, a high-level planning system that is aware of current task requirements may have good reasons to modify weights. If, for example, the high-level planning system can infer that energy consumption is irrelevant because it knows that there will certainly be enough time to recharge in-between tasks, it can simply set the importance of saving energy w_E to $w_E = 0$.

6.2.2 Utility Pertaining to Execution Success

In order to compute *expected* utility of success $u_S(x, y)$, the combined grasp success probability $p(x, y)$ is multiplied with importance of success w_S.

$$u_S(x, y) = p(x, y) \cdot w_S \tag{6.4}$$

6.2.2.1 Combined Grasp Success Probability

For every grid cell, the combined grasp success probability represents the probability that the robot will be able to successfully grasp the target object when the grasp is performed from within that grid cell. It is computed as the product of basic grasp success probability $p_S(x, y)$ and the probability that the grid cell is unoccupied.

$$p(x, y) = p_S(x, y) \cdot p_U(x, y) \tag{6.5}$$

Probabilistically, equation 6.5 can be read as

$$\begin{aligned} P(\textit{CombinedSuccess}) &= P(\textit{Success}, \textit{Unoccupied}) \\ &= P(\textit{Success} \mid \textit{Unoccupied}) \cdot P(\textit{Unoccupied}) \end{aligned}$$

because $P(\textit{Success}, \neg\textit{Unoccupied}) = 0$.

6.2.2.2 Importance of Success

The importance of success w_S is defined as the (saved) time that is typically required to successfully redo the task. It is evaluated by the high-level planner and transferred to the ARPLACE framework as a parameter. Intuitively, the importance of success is the high-level planner's measure of how important it is to succeed in performing the subsequent manipulation action. We present three robot tasks and describe how the importance of success could be reasonably set by the high level planner.

A) Grasp an empty plastic cup
B) Grasp a plastic cup filled with juice
C) Grasp an empty ceramic cup

For task A), $w_S = 30$s because in case of failure the robot needs 10s to reposition its base and 20s to perform an additional grasping action. For task B), $w_S = 330$s because failure implies that juice will be spilled, and the robot thus estimates the time for redoing the grasp plus an additional five minutes to clean up. For task C), $w_S = 2520$s because in case of failure the cup will break with probability 0.5, requiring to spend 60 minutes in order to go to a supermarket and buy a new cup which costs 4 Euros. Assuming that 24 minutes are required to earn 4 Euros, this leads to a total of $0.5 \cdot (60\text{min} + 24\text{min}) = 42\text{min}$.

6.2.3 Utility Pertaining to Execution Time

In this section, we present a method for computing the utility pertaining to execution time $u_T(x, y)$. This enables robots to take action duration into account whenever the expected utility of grasp success is not the only concern.

Please keep in mind that ARPLACEs find globally optimal manipulation places, and therefore compute a complete mapping from grid cells to utilities. In order to consider the utility pertaining to execution time, the robot has to compute - for every grid cell - the execution time that it can expect for moving to the grid cell and performing the grasp. We found that the time required to perform the grasp itself varies only marginally and is dominated by the time required to travel to a particular target position.

A viable but computationally expensive approach to estimate travel time would be to run the robot's motion planner in order to find trajectories to all relevant grid cells and subsequently estimate the time that is required for each trajectory. However, this is very time consuming because we would have to solve a separate motion query for every grid cell. Because our robot uses Player's Wavefront planner, which performs global path planning based on finding shortest paths, we approximate Wavefront's behaviour by using Dijkstra's algorithm. This enables us to find trajectories to all relevant grid cells in a single run and significantly reduces runtime. In the following, we present an accurate motion model that is tailored towards our B21r mobile robot. The motion model allows to precisely estimate travel time for given trajectories.

6.2.3.1 Navigation Time Model

The B21r mobile robot uses a synchro drive which allows the robot to move forward, move backward, and turn to the left and right at any time. These motion opportunities are depicted in the left image of Figure 6.4. The robot can move to any of the eight neighboring grid cells, as long as its orientation is correct. It can change its orientation by turning 45° in either direction. For example, if the robot is initially facing left, as shown in the left image of Figure 6.4, it can move upward by either turning right by 45° twice and moving forward, or turning left by 45° twice and moving backward. The former alternative leads to the state that is framed blue in the right image of Figure 6.4.

To associate travel times to trajectories we measured the robot's translational velocities for moving forward ($v_{\textit{for}}$) and backward ($v_{\textit{back}}$), as well as the robot's rotational velocities for turning to the left ($v_{\textit{left}}$) and right ($v_{\textit{right}}$):

- $v_{\textit{for}} = v_{\textit{back}} = 0.1 \ \frac{\mathrm{m}}{\mathrm{s}}$
- $v_{\textit{left}} = v_{\textit{right}} = 6 \ \frac{\circ}{\mathrm{s}}$

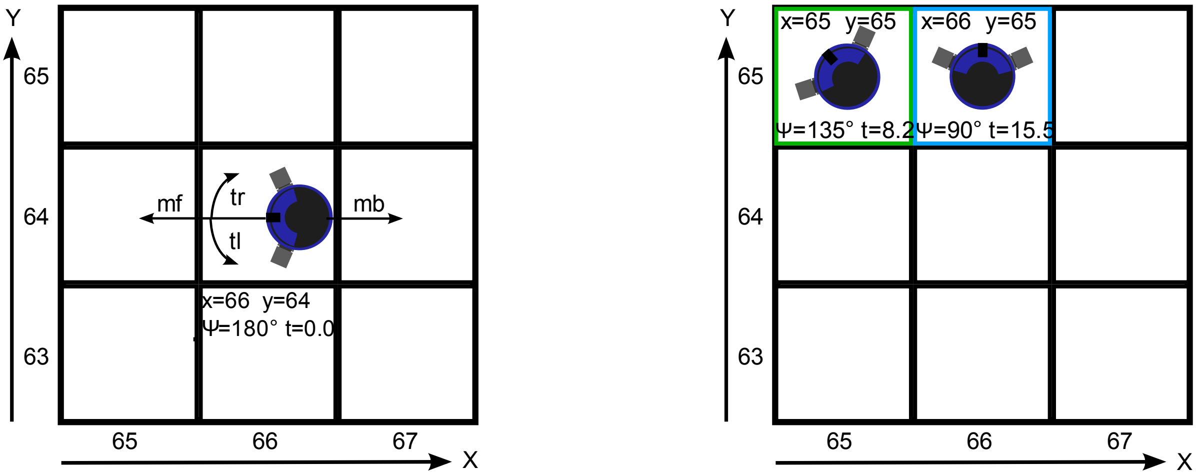


FIGURE 6.4 Left: Illustration of our robot's motion model. Each grid cell measures 5cm × 5cm. Atomic movement actions: move forward (mf), move backward (mb), turn right (tr), turn left (lf). The robot state is defined by the values x, y, and ψ. The estimated travel time is denoted by t. Right: The robot's estimated travel time for moving from its initial position to the position with the blue frame is 15.5s (15s for turning right 90° and 0.5s for moving forward 5cm). The green frame is reachable within 8.2s by turning right 45° and moving forward 7.1cm.

The motion model implicitly defines the search tree of our path planning problem. As shown in Figure 6.5, every node is connected to four successor nodes representing the states reached through the four atomic movements.

By applying Dijkstra's algorithm, we obtain travel times to all unoccupied grid cells. The travel time to occupied grid cells is set to ∞. Eight time values are stored for every grid cell, because we consider eight angular orientations of the robot. The left image of Figure 6.6 visualizes estimated travel times from the robot's current base position to all grid cells. For every grid cell, the time value that is plotted corresponds to the robot facing right. For comparison, the right image of Figure 6.6 visualizes estimated travel times where the robot finally faces down.

Several abrupt color changes can be observed in both images, with the green line in the right image of Figure 6.6 being the most obvious. The reason for this green line is that the turning speed of our B21r robot is slow. It is important to keep in mind that the robot's initial position is $rob = \langle x_{\text{rob}}, y_{\text{rob}}, \psi_{\text{rob}} \rangle = \langle 4.5\text{m}, 3.9\text{m}, \frac{3}{2}\pi \rangle$. So the robot is facing down. The grid cells of the right image of Figure 6.6 represent travel times where the robot also faces down at the goal position ($\psi'_{\text{rob}} = \frac{3}{2}\pi$). Therefore, when moving to any target position where $x'_{rob} = 4.5\text{m}$, no turning motion is required. Estimated travel time for moving to target pose $rob' = \langle 4.5\text{m}, 2.4\text{m}, \frac{3}{2}\pi \rangle$ for example is 15.0s, because the robot moves forward by 1.5m with a velocity of $0.1\frac{\text{m}}{\text{s}}$. For any target pose where $x'_{rob} \neq 4.5\text{m}$, *at least* two turning motions

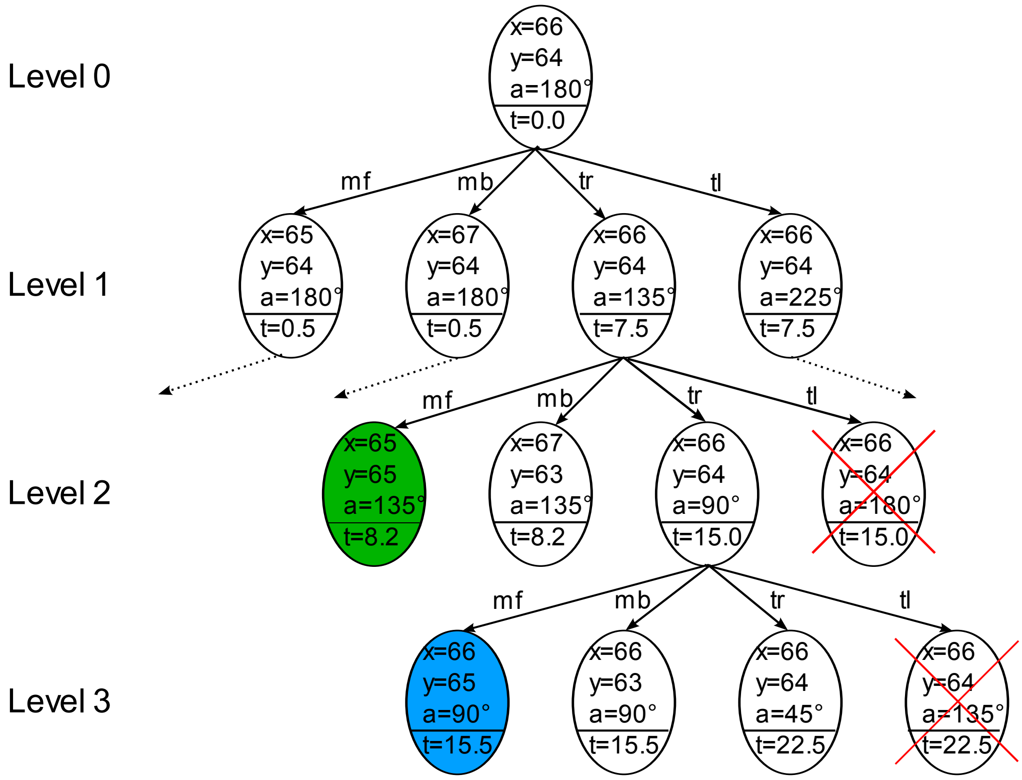


FIGURE 6.5 Search tree for estimating travel time. The green and blue nodes correspond to the robot states that are shown in the right image of Figure 6.4. The subtrees that are rooted at nodes that are crossed out are pruned during search, as their root nodes correspond to states that have been explored previously.

of 45° are required. One turning motion to the left or right for enabling the robot to move diagonal, and one turning motion into the opposite direction to make the robot face down again. Estimated travel time for moving to target position $rob' = \langle 4.7\mathrm{m}, 3.7\mathrm{m}, \frac{3}{2}\pi \rangle$ would be 17.8s, although the grid cell is much closer. Generally, abrupt changes in estimated travel time happen at neighboring grid cells where one grid cell requires more turning motions than the other one.

6.2.3.2 Utility Pertaining to Execution Time

In order to compute the utility pertaining to execution time, utility of time $u_T(x, y)$ is multiplied with the importance of time w_T. Utility of time is derived from the estimated travel time $t(x, y)$ as computed in the last section. Because lower estimated travel time should have higher utility, we set the utility of time to

$$u_T(x, y) = -t(x, y) \tag{6.6}$$

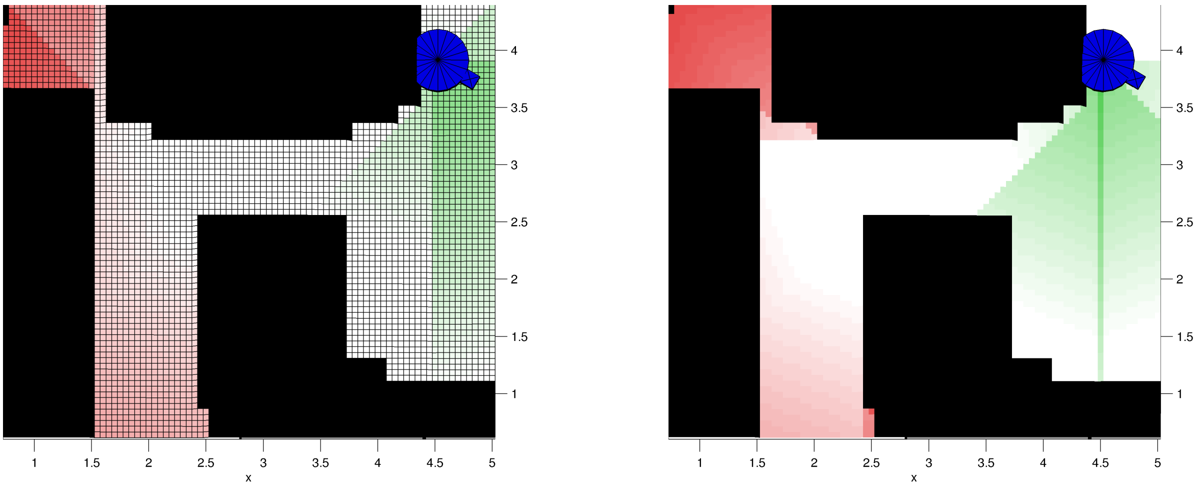


FIGURE 6.6 Estimated travel time from the robot's inital position to every grid cell. Green indicates low travel time, white and red successively longer times. Unreachable grid cells are black. Obstacles have been grown by the robot's radius so that the robot can be considered as a point for collision checks. Left: Travel times for robot facing right at the goal grid cell. Right: Travel Times for robot facing down. Grid borders are omitted for clarity.

In the left image of Figure 6.6, the utility of time ranges from -0.0s for the grid cell where the robot is currently located to -84.3s for the farthest grid cell. Grid cells that are occupied by an obstacle have a utility value of $-\infty$s.

The importance of time is chosen by the high level planner according to the importance that is attributed to the goal of performing the task as quickly as possible. Small values of w_T result in a tendency to prefer manipulation places that maximize combined grasp success probability, even if they are far away. High values of w_T result in a tendency to save time and and prefer manipulation places nearby, even if combined grasp success probability is low.

6.3 Evaluation

In this evaluation we consider a concrete task scenario and illustrate the effect of different problem parameterizations on the choice of the manipulation place. According to equation 6.1, the expected utility of utility-based ARPLACES is computed as follows when considering grasp success probability and execution time.

$$u(x, y) = p(x, y) \cdot w_s + u_t(x, y) \cdot w_t$$

The four parameters that determine expected utility are combined grasp success probability $p(x, y)$, importance of success w_s, utility of time $u_t(x, y)$, and the importance of time w_t.

While $p(x, y)$ and $u_t(x, y)$ are computed by the ARPLACE framework, w_s and w_t are computed by a high-level planning system in order to reflect if grasp quality or execution time is more important for the task at hand. The parameters w_s and w_t are passed by the high-level planning system to the ARPLACE framework. Utility-based ARPLACEs then find optimal manipulation places for trading-off grasp success probability and execution time.

We start our evaluation by presenting the default evaluation scenario in section 6.3.1, where every of the above mentioned four parameters is set to a default value. In sections 6.3.2 to 6.3.5 we change one parameter while keeping the other three parameters fix, and observe the impact on the resulting ARPLACE utility distribution. To get an intuition about how the utility-based ARPLACE and the ARPLACE that is based on grasp success probability differ, we compare the manipulation places that are proposed by these approaches. We finish our evaluation in section 6.3.6 where a series of experiments is performed in order to analyze the average trade-off between grasp success probability and execution time.

6.3.1 Default Scenario

The images of Figure 6.7 depict the default evaluation scenario. The pose of the target object with respect to the world frame is $obj = \langle 3.05\text{m}, 1.9\text{m}, -\frac{\pi}{2}\rangle$. The initial pose of the robot is $rob = \langle 4.5\text{m}, 3.9\text{m}, -\frac{\pi}{2}\rangle$. The positional uncertainties of the robot into its base position and the cup's pose are $\sigma_{x_{\text{rob}}} = \sigma_{y_{\text{rob}}} = \sigma_{x_{\text{obj}}} = \sigma_{y_{\text{obj}}} = 0.04\text{m}$.

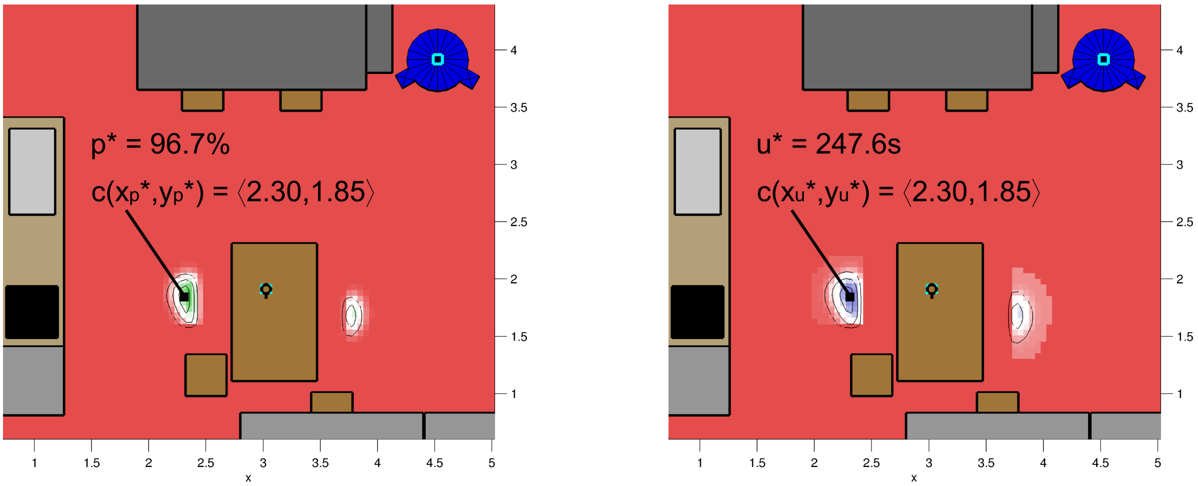


FIGURE 6.7 Default evaluation scenario. Left: ARPLACE based on grasp success probability. Right: Utility-based ARPLACE. Grid cells with high utility are colored blue in order to distinguish utility plots from probability plots.

The left image of Figure 6.7 depicts an ARPLACE that is based on (combined) grasp success probability. p^* refers to the maximum probability value of the probability distribution, and

(x_{p^*}, y_{p^*}) is the grid cell with maximal grasp success probability. (x_{p^*}, y_{p^*}) is labeled black in the left image of Figure 6.7. $c(x_{p^*}, y_{p^*})$ refers to the coordinates of the center of the grid cell with maximal grasp success probability. It can be seen, that for the probability-based ARPLACE $p^* = 96.7\%$ and $c(x_{p^*}, y_{p^*})$ is $\langle 2.30\text{m}, 1.85\text{m}\rangle$.

Please note that the target object is reachable from the left and right table side, and therefore a multi-modal ARPLACE distribution emerges. Because the target object's pose is slightly more on the left side of the table, grasp success probability is generally higher in the left ARPLACE cluster. However, it is obvious that the cluster on the right table side is closer to the robot's initial position and can be reached faster.

The right image of Figure 6.7 depicts a utility-based ARPLACE. Please note that grid cells with high utility values are colored blue in order to make utility plots distinguishable from probability plots. The robot expects the target object to be a cup that is filled with juice, so the importance of success is $w_S = 330\text{s}$. There are no time constraints, so the robot uses the default value for importance of time and sets $w_t = 1.0$. u^* refers to the maximal utility value of the utility distribution. (x_{u^*}, y_{u^*}) is the grid cell with maximal utility and labelled black in the right image of Figure 6.7. $c(x_{u^*}, y_{u^*})$ refers to the coordinates of the center of the grid cell with maximal utility. It can be seen, that for the utility-based ARPLACE $u^* = 247.6\text{s}$. It was computed as

$$u^* = 0.967 \cdot 330.0\text{s} + 1.0 \cdot (-71.47\text{s}) = 247.6\text{s}$$

The above equation shows that - for the current task - the utility distribution is clearly dominated by grasp success probability. $c(x_{u^*}, y_{u^*})$ is $\langle 2.30\text{m}, 1.85\text{m}\rangle$. In this scenario the probability-based and utility-based ARPLACE propose the same manipulation place because $(x_{p^*}, y_{p^*}) = (x_{u^*}, y_{u^*})$. This is intuitive, as the risk of performing a bad grasp, hitting the cup, and spilling the juice over the ground is not worth trying to save execution time. Especially when execution time is not particularly important, which is not the case for this task because $w_T = 1.0$.

6.3.2 Impact of Importance of Success

In this section we evaluate the impact of the importance of success w_S on the utility distribution. Therefore we change w_S while keeping the other values fix at their default values, so the importance of time is $w_T = 1.0$, the pose of the target object is $obj = \langle 3.05\text{m}, 1.9\text{m}, \frac{3}{2}\pi\rangle$, and the initial position of the robot is $rob = \langle 4.5\text{m}, 3.9\text{m}, \frac{3}{2}\pi\rangle$.

Importance of success is computed by the high-level planner in order to reflect the importance that the manipulation action succeeds. Section 6.2.2.2 presented three tasks and showed how the corresponding values for u_S are computed. The results were $w_S = 30\text{s}$ for task A) of grasping an empty plastic cup, $w_S = 330\text{s}$ for task B) of grasping a plastic cup filled with juice, and $w_S = 2520\text{s}$ for task C) of grasping an empty ceramic cup. Figure 6.8 depicts utility distributions for performing task A) in the left image and task C) in the right image.

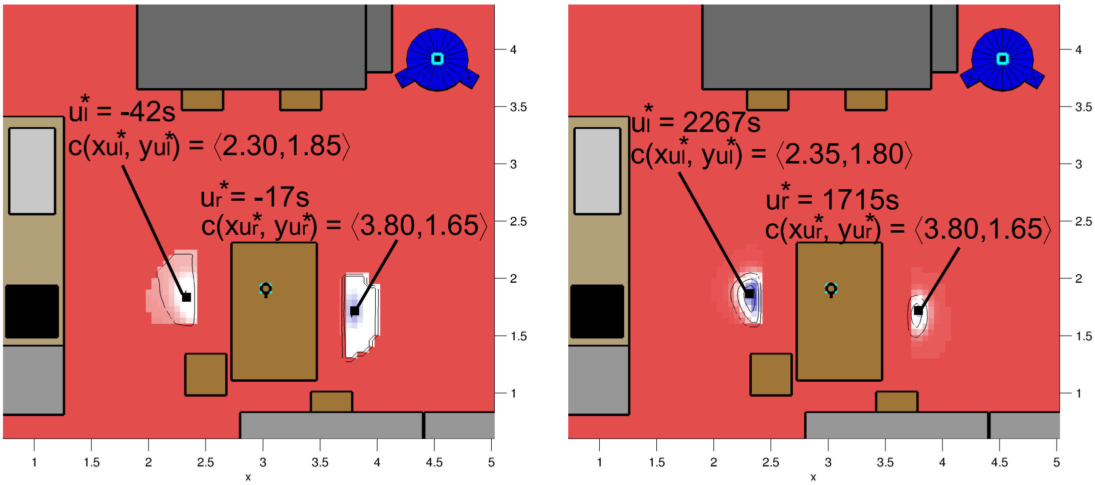


FIGURE 6.8 Utility distributions for different values of w_S. Left: Utility distribution for performing task A), where $w_S = 30\text{s}$ (experiment 1 in Figure 6.9). Right: Utility distribution for performing task C), where $w_S = 2520\text{s}$ (experiment 3).

The plots visualize the maximum utility value of the left ARPLACE cluster which is denoted by u_l^*, and the maximum utility value of the right ARPLACE cluster which is denoted by u_r^*. It can be seen that for task A) where grasp success is relatively unimportant, $u_r^* = -17\text{s}$ and $u_l^* = -42\text{s}$. Because $u_r^* > u_l^*$ the ARPLACE proposes to perform the grasping action from the right table side and proposes $c(x_{u^*}, y_{u^*}) = \langle 3.80\text{m}, 1.65\text{m}\rangle$ as manipulation place. The explanation is that the robot prefers to perform the grasp from a nearby position in order to save execution time. For task C) where grasp success is more important $u_r^* = 1715\text{s}$ and $u_l^* = 2267\text{s}$. Because $u_l^* > u_r^*$ the ARPLACE proposes to perform the grasping action from the left table side and proposes $c(x_{u^*}, y_{u^*}) = \langle 2.35\text{m}, 1.80\text{m}\rangle$ as manipulation place. This requires the robot to move around the table in order to have the advantage of higher grasp success probability.

Figure 6.9 depicts a table with all important values for computing expected utility of performing task A), B), and C). The plots that are depicted in Figure 6.8 refer to the data that is specified in experiment 1 (left image) and 3 (right image) of the table. For each task, the maximum utility value u^* is shown for the ARPLACE cluster on the left and right table side. As expected, grasp success probability is higher for grid cells on the left table side with maximum probabilities ranging from 93% to 97% compared to 65% to 69% on the right table side.

Exper	L/R	c(x_{u^*}, y_{u^*})	u^*	$p(x_{u^*}, y_{u^*})$	w_S	w_T	$u_T(x_{u^*}, y_{u^*})$
1	L	$\langle 2.30\text{m}, 1.85\text{m}\rangle$	-42s	0.97	30s	1.0	-71.0s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	-17s	0.67	30s	1.0	-36.9s
2	L	$\langle 2.35\text{m}, 1.80\text{m}\rangle$	238s	0.94	330s	1.0	-71.3s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	177s	0.65	330s	1.0	-36.9s
3	L	$\langle 2.35\text{m}, 1.80\text{m}\rangle$	2267s	0.93	2520s	1.0	-71.3s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	1715s	0.69	2520s	1.0	-36.9s

FIGURE 6.9 Expected utilities for manipulation tasks A) - C). Varying the manipulation task leads to different values of w_S (colored red). Abbreviations: L/R: values in this row refer to utility distribution on the L(eft)/R(ight) table side; $c(x_{u^*}, y_{u^*})$: center of grid cell with maximum expected utility; u^*: maximum expected utility; $p(x_{u^*}, y_{u^*})$: grasp success probability at grid cell (x_{u^*}, y_{u^*}); w_S: importance of success; w_T: importance of time; $u_T(x_{u^*}, y_{u^*})$: utility of time at grid cell (x_{u^*}, y_{u^*}).

There are slight variations in grasp success probability because of the random sampling steps that take uncertainties into robot position and object pose into account. On the other side, estimated travel time to the right table side is shorter with 36.9s compared to approximately 71.0s to manipulation places on the left table side.

6.3.3 Impact of Importance of Time

In this section we evaluate the importance of time w_T on the ARPLACE utility distribution. The other parameters are set to their default values. Importance of time is computed by the high-level planner and represents the importance to perform the manipulation task as quickly as possible. High values of w_T represent the need for short execution time, while low values represent that the task is not urgent. Figure 6.10 depicts the utility distributions for $w_T = 0.0$ in the left image and $w_T = 3.0$ in the right image.

It can be seen that for $w_T = 0.0$ the ARPLACE proposes to perform the manipulation action from the left table side, because $u_l^* = 28\text{s}$ and $u_r^* = 20\text{s}$. The reason is that a value of $w_T = 0.0$ means that execution time is completely irrelevant, making grasp success probability the only criterion for choosing a manipulation place. When the high-level planner computed the importance of time to be $w_T = 3.0$, then performing the task as quickly as possible is important. As a result, the ARPLACE utility distribution changes significantly which can be seen when comparing the images of Figure 6.10. Now $u_l^* = -186\text{s}$ and $u_r^* = -88\text{s}$. Because $u_r^* > u_l^*$ the ARPLACE framework proposes to move to base position $\langle 3.80\text{m}, 1.75\text{m}\rangle$ which trades grasp success probability for saving time.

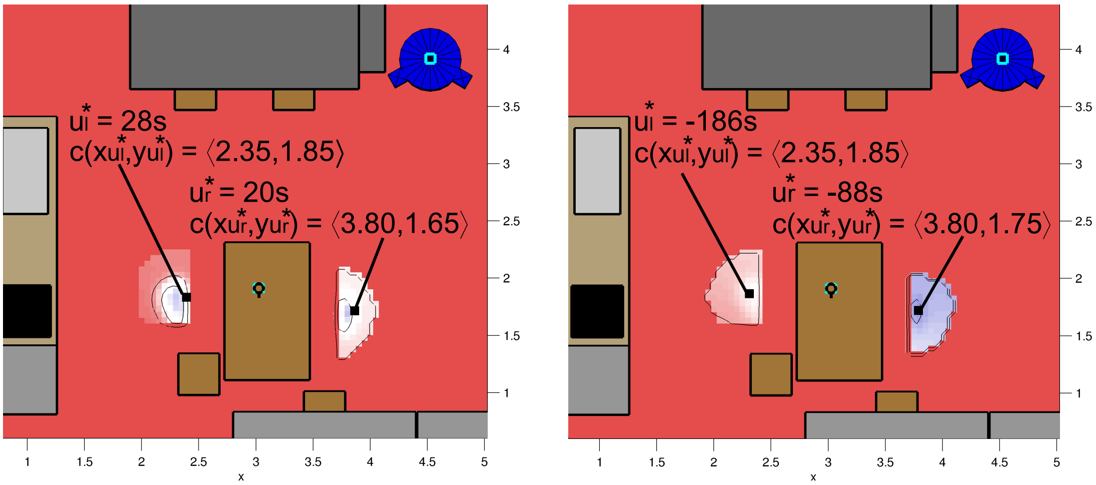


FIGURE 6.10 Utility distributions for different values of w_t. Left: Utility distribution for $w_T = 0.0$ (experiment 1 in Figure 6.11). Right: Utility distribution for $w_T = 3.0$ (experiment 4).

The table in Figure 6.11 depicts a detailed overview of all relevant values of eight experiments, where the plots in Figure 6.10 refer to experiment number 1 and 4. It can be seen, that $u_l^* > u_r^*$ is only valid for $w_T = 0.0$ (experiment 1). For all other values of w_T the utility-based ARPLACE proposes to move to a base position on the right table side. This can be attributed to the small utility of success of task A). That is why we performed additional experiments to further analyse the importance of time. We performed task B) where $w_S = 330\text{s}$. In this case $u_l^* > u_r^*$ for values up to $w_T = 1.0$ (experiment 6), and even when $w_T = 3.0$ u_l^* is only slightly smaller than u_r^* (experiment 7).

A very interesting result is that for all experiments so far the utility-based ARPLACE proposed manipulation places at or near $\langle 3.80\text{m}, 1.65\text{m}\rangle$ when $u_r^* > u_l^*$, and manipulation places at or near $\langle 2.30\text{m}, 1.85\text{m}\rangle$ when $u_l^* > u_r^*$. This is not the case for experiment 5, where $w_T = 10.0$. It is so important to save time in this experiment that the utility-based ARPLACE proposes the nearest available grid cell that has a grasp success probability of more than 0%. Grasp success probability itself is rendered unimportant. Figure 6.12 depicts the corresponding ARPLACE from a top-down view and from an isometric perspective.

The proposed base position is $\langle 3.85\text{m}, 2.10\text{m}\rangle$ with an estimated travel time of 32.7s which is smaller than the estimated travel times of about 37.0s for the manipulation places that have been proposed so far. Grasp success probability is 0.002%. The grid cell with maximal utility in the left ARPLACE cluster is also the one that is closest to the robot $\langle 2.35\text{m}, 2.25\text{m}\rangle$. This is an extreme example, but demonstrates that w_T has to be chosen with care. When performing task B) that has a higher utility of success and setting $w_T = 10.0$, then the base position $\langle 3.80\text{m}, 1.70\text{m}\rangle$ is proposed that is still on the right table side, but less extreme with a grasp

Exper	L/R	c(x_{u^*}, y_{u^*})	u^*	$p(x_{u^*}, y_{u^*})$	w_S	w_T	$u_T(x_{u^*}, y_{u^*})$
1	L	$\langle 2.35\text{m}, 1.85\text{m}\rangle$	28s	0.93	30s	0.0	-71.3s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	20s	0.66	30s	0.0	-37.4s
2	L	$\langle 2.35\text{m}, 1.85\text{m}\rangle$	7s	0.95	30s	0.3	-71.3s
	R	$\langle 3.80\text{m}, 1.70\text{m}\rangle$	9s	0.68	30s	0.3	-36.9s
3	L	$\langle 2.35\text{m}, 1.90\text{m}\rangle$	-42s	0.95	30s	1.0	-70.8s
	R	$\langle 3.80\text{m}, 1.70\text{m}\rangle$	-16s	0.66	30s	1.0	-36.9s
4	L	$\langle 2.35\text{m}, 1.85\text{m}\rangle$	-186s	0.94	30s	3.0	-71.3s
	R	$\langle 3.80\text{m}, 1.75\text{m}\rangle$	-88s	0.66	30s	3.0	-36.4s
5	L	$\langle 2.35\text{m}, 2.25\text{m}\rangle$	-672s	0.00002	30s	10.0	-67.3s
	R	$\langle 3.85\text{m}, 2.10\text{m}\rangle$	-326s	0.00002	30s	10.0	-32.7s
6	L	$\langle 2.35\text{m}, 1.85\text{m}\rangle$	244s	0.96	330s	1.0	-71.3s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	193s	0.69	330s	1.0	-37.4s
7	L	$\langle 2.35\text{m}, 1.90\text{m}\rangle$	100s	0.95	330s	3.0	-70.8s
	R	$\langle 3.80\text{m}, 1.70\text{m}\rangle$	104s	0.65	330s	3.0	-36.9s
8	L	$\langle 2.35\text{m}, 1.90\text{m}\rangle$	-398s	0.94	330s	10.0	-70.8s
	R	$\langle 3.80\text{m}, 1.70\text{m}\rangle$	-147s	0.67	330s	10.0	-36.9s

FIGURE 6.11 Experiments 1-5 show expected utilities for performing task A) and varying w_t. Experiments 6-8 show expected utilities for performing task B) and varying w_T.

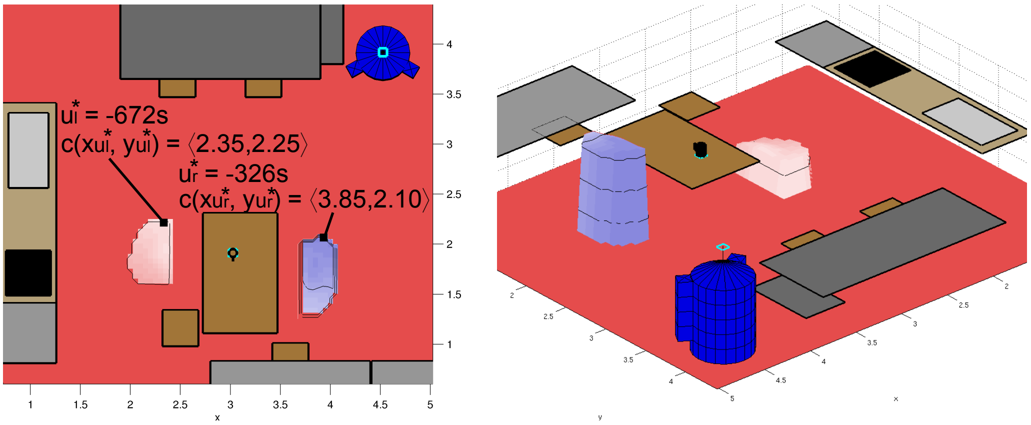


FIGURE 6.12 ARPLACE utility distributions when the task is urgent. The utility distributions are depicted from a top-down view (left image) and from an isometric perspective (right image).

success probability of 67% (experiment 8).

6.3.4 Impact of Initial Robot Position

In this section we evaluate the impact of the robot's initial base position on the utility distribution. Changing the robot's initial position leads to a change of the grid cell's utility of time $u_T(x, y)$. The other parameters are set to their default values. The assumption is that when the robot's initial position is on the right table side, then it will prefer to grasp the target object from the right table side, while grid cells on the left table side are prefered when the initial base position is on the left table side. Experiments are performed from four different initial base positions that can be seen in the left image of Figure 6.13.

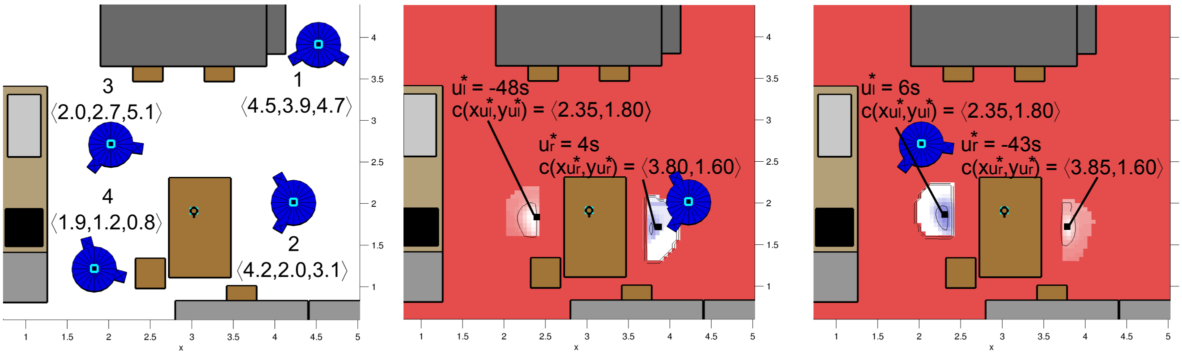


FIGURE 6.13 Left: Different initial base positions of the robot. The single number next to a initial position refers to the experiment number in which it is used. The numbers in brackets specify coordinates of the initial position. Center: Utility distribution for initial position of experiment 2. Right: Utility distribution for initial position of experiment 3.

The first initial base position is the robot's default initial position. The second initial position is near to the probability cluster on the right table side, and the third and fourth initial positions are near to the probability cluster on the left table side. The plots in Figure 6.14 depict estimated travel time from all four initial robot positions. This information is useful in the following discussion.

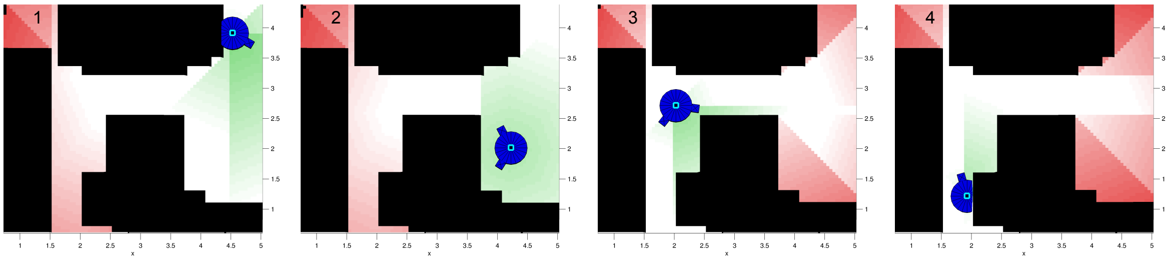


FIGURE 6.14 Time plots for different initial robot positions. Each plot is labeled with the corresponding experiment number. For every grid cell, the time value is plotted where the robot faces right.

The center and right image of Figure 6.13 depict resulting utility distributions for experiment 2 and 3. It can be seen that our assumption holds. When the robot's initial base position is $rob = \langle 4.2\text{m}, 2.0\text{m}, \pi \rangle$, which is close to the right table side, then $u_r^* = 4\text{s}$ and $u_l^* = -48\text{s}$. The main reason why $u_r^* > u_l^*$ is that estimated travel time is significantly lower with 17.5s to $c(x_{u_r^*}, y_{u_r^*})$ compared to 76.5s to $c(x_{u_l^*}, y_{u_l^*})$. When the robot's initial base position is $rob = \langle 2.0\text{m}, 2.7\text{m}, 5.1\text{rad} \rangle$, which is close to the left table side, then $u_r^* = -43\text{s}$ and $u_l^* = 6\text{s}$. In this case, the estimated travel time to $c(x_{u_r^*}, y_{u_r^*})$ is 63.5s compared to 22.0s for moving to $c(x_{u_l^*}, y_{u_l^*})$.

The table in Figure 6.15 depicts a detailed overview of the values of the experiments. In experiment 1, the utility-based ARPLACE proposes the manipulation place $\langle 3.80\text{m}, 1.65\text{m} \rangle$, which is at the right table side. The reason is that the estimated travel time to u_r^* is shorter than to u_l^* (36.9s compared to 70.8s), which outweighs the increase of grasp success probability (66% compared to 94%). Experiments 2 and 3 were already discussed above. The utility-based ARPLACE unsurprisingly proposes to perform the manipulation action from the left table side in experiment 4 because estimated travel time is shorter and grasp success probability is higher.

Exper	L/R	c(x_{u^*}, y_{u^*})	u^*	$p(x_{u^*}, y_{u^*})$	u_S	w_T	$u_T(x_{u^*}, y_{u^*})$
1	L	$\langle 2.35\text{m}, 1.85\text{m} \rangle$	-42s	0.94	30s	1.0	-70.8s
	R	$\langle 3.80\text{m}, 1.65\text{m} \rangle$	-17s	0.66	30s	1.0	-36.9
2	L	$\langle 2.35\text{m}, 1.80\text{m} \rangle$	-48s	0.94	30s	1.0	-76.5s
	R	$\langle 3.80\text{m}, 1.60\text{m} \rangle$	4s	0.70	30s	1.0	-17.5s
3	L	$\langle 2.35\text{m}, 1.80\text{m} \rangle$	6s	0.93	30s	1.0	-22.0s
	R	$\langle 3.85\text{m}, 1.60\text{m} \rangle$	-43s	0.67	30s	1.0	-63.5s
4	L	$\langle 2.35\text{m}, 1.80\text{m} \rangle$	2s	0.93	30s	1.0	-26.4s
	R	$\langle 3.80\text{m}, 1.65\text{m} \rangle$	-60s	0.67	30s	1.0	-80.0s
5	L	$\langle 2.35\text{m}, 1.85\text{m} \rangle$	237s	0.95	330s	1.0	-76.0s
	R	$\langle 3.80\text{m}, 1.65\text{m} \rangle$	209s	0.69	330s	1.0	-17.2s

FIGURE 6.15 Experiments 1-4 show expected utilities for performing task A) from different initial base positions. Experiment 5 shows expected utility for performing task B) from the same initial base position as in experiment 2.

The robot's initial base position in experiment 5 is the same as in experiment 2, but this time task B) is performed instead of task A). Now the ARPLACE framework proposes to perform the manipulation action from the left table side although starting at the right table side. Experiment 5 shows that it is advantageous to maximize grasp success probability when importance of success increases even when estimated travel time is longer.

6.3.5 Impact of Object Pose

In this section we evaluate the impact of the target object's initial pose on the utility distribution. Changing the object's initial pose leads to a change of grasp success probability. The other parameters are set to their default values. The assumption is that when the object's initial pose is closer to the right table side, then the robot will prefer to grasp the target object from the right table side, while grid cells on the left table side are prefered when the object is closer to the left table side.

Experiments are performed for five different cup poses that can be seen in the left image of Figure 6.16. Please note that the cup poses are specified with respect to the table frame. The poses start from being closer to the left table side, and stop at being closer to the right table side. The second cup pose is the cup's default pose. The center and right image of Figure 6.16 depict resulting utility distributions for experiment 3 and 5, and the table in Figure 6.17 depicts a detailed overview of all values of the experiments.

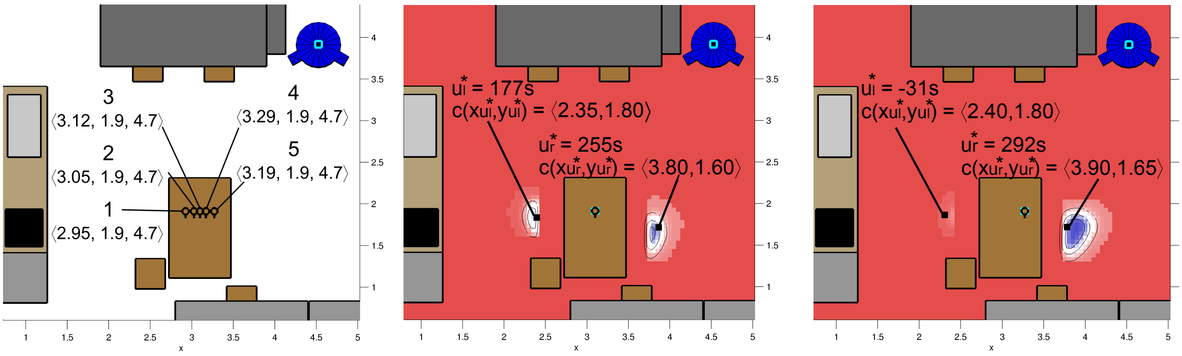


FIGURE 6.16 Left: Different initial poses of the target object. The single number next to an object pose refers to the corresponding experiment number. The numbers in brackets specify the pose. Center: Utility distribution for experiment 3, where the distance of the cup to each table side is equal. Right: Utility distribution for experiment 5.

The results show that the assumption holds. In experiment 1, the utility-based ARPLACE proposes to perform the manipulation from position $\langle 2.30\mathrm{m}, 1.80\mathrm{m}\rangle$ at the left table side, because the $u_l^* = 257\mathrm{s}$ is higher than $u_r^* = 5\mathrm{s}$. The reason is that grasp success probability of the grid cell $(x_{u_l^*}, y_{u_l^*})$ is higher than grasp success probability of grid cell $(x_{u_r^*}, y_{u_r^*})$. More specific, $p(x_{u_l^*}, y_{u_l^*}) = 0.99$ and $p(x_{u_r^*}, y_{u_r^*}) = 0.13$. This compensates the higher estimated travel time of 71.5s for traveling to $c(x_{u_l^*}, y_{u_l^*})$ which is 71.5s compared to 36.9s for traveling to $c(x_{u_r^*}, y_{u_r^*})$. The same is the case in experiment 2 where grasp success probability is 96% for $(x_{u_l^*}, y_{u_l^*})$ and 67% for $(x_{u_r^*}, y_{u_r^*})$. Experiment 3 is the borderline case, where the cup is positioned so that the distance to the right and left table side is equal with 0.3725m. Therefore, grasp success probability is similar with 75% for $(x_{u_l^*}, y_{u_l^*})$ compared to 88% for $(x_{u_r^*}, y_{u_r^*})$. In

Exper	L/R	$c(x_{u^*}, y_{u^*})$	u^*	$p(x_{u^*}, y_{u^*})$	u_S	w_T	$u_T(x_{u^*}, y_{u^*})$
1	L	$\langle 2.30\text{m}, 1.80\text{m}\rangle$	257s	0.99	330s	1.0	-71.5s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	5s	0.13	330s	1.0	-36.9s
2	L	$\langle 2.30\text{m}, 1.85\text{m}\rangle$	247s	0.96	330s	1.0	-71.0s
	R	$\langle 3.80\text{m}, 1.65\text{m}\rangle$	183s	0.67	330s	1.0	-36.9s
3	L	$\langle 2.35\text{m}, 1.80\text{m}\rangle$	177s	0.75	330s	1.0	-71.3s
	R	$\langle 3.80\text{m}, 1.60\text{m}\rangle$	255s	0.88	330s	1.0	-37.4s
4	L	$\langle 2.40\text{m}, 1.80\text{m}\rangle$	107s	0.54	330s	1.0	-71.1s
	R	$\langle 3.90\text{m}, 1.65\text{m}\rangle$	283s	0.96	330s	1.0	-36.5s
5	L	$\langle 2.40\text{m}, 1.80\text{m}\rangle$	-31s	0.12	330s	1.0	-71.1s
	R	$\langle 3.90\text{m}, 1.65\text{m}\rangle$	292s	0.99	330s	1.0	-36.5s

FIGURE 6.17 Expected utilities for performing task B) where the target object is located at different locations.

such a situation, the robot will choose whatever position can be reached faster. In this experiment the robot's intitial base position is to the right of the table, which leads to manipulation places with higher utility values on the right table side with $u_r^* = 255\text{s}$ compared to $u_l^* = 177\text{s}$.

For experiments 4 and 5 the preference for manipulation places on the right table side increases, because the cup's distance to the right table side decrease, and grasp success probability increases accordingly. Actually, grasp success probability in experiments 4 and 5 are mirror images to experiments 1 and 2. This can be seen when studying the probability values $p(x_{u_l^*}, y_{u_l^*})$ in Figure 6.17. It is particularly obvious when watching Figure 6.18 where a plot of grasp success probability is depicted for every experiment.

6.3.6 Average Behavior

In the second part of the evaluation we compare the newly introduced utility-based ARPLACEs (ARPLACE$_{\text{UTIL}}$) to ARPLACEs that are based on grasp success probability (ARPLACE$_{\text{PROB}}$). We considered the following 21 target object poses (obj) and 21 initial robot positions (rob), both equally distributed along a line segment parallel to the x-axis.

$obj \in \{\langle 2.92\text{m}, 1.9\text{m}, \frac{3}{2}\pi\rangle, \langle 2.94\text{m}, 1.9\text{m}, \frac{3}{2}\pi\rangle, .., \langle 3.32\text{m}, 1.9\text{m}, \frac{3}{2}\pi\rangle\}$

$rob \in \{\langle 1.82\text{m}, 2.9\text{m}, \frac{3}{2}\pi\rangle, \langle 1.95\text{m}, 2.9\text{m}, \frac{3}{2}\pi\rangle, .., \langle 4.42\text{m}, 2.9\text{m}, \frac{3}{2}\pi\rangle\}$

Four out of 21 initial robot positions and six out of 21 object poses are depicted in Figure 6.19. For each pair of positions and each of the tasks A) - C), we computed ARPLACE$_{\text{UTIL}}$ as well as ARPLACE$_{\text{PROB}}$, resulting in a total of $21 \cdot 21 \cdot 3 \cdot 2 = 2646$ ARPLACE computations.

Figure 6.20 summarizes the results in a table, and Figure 6.21 visualizes them in plots.

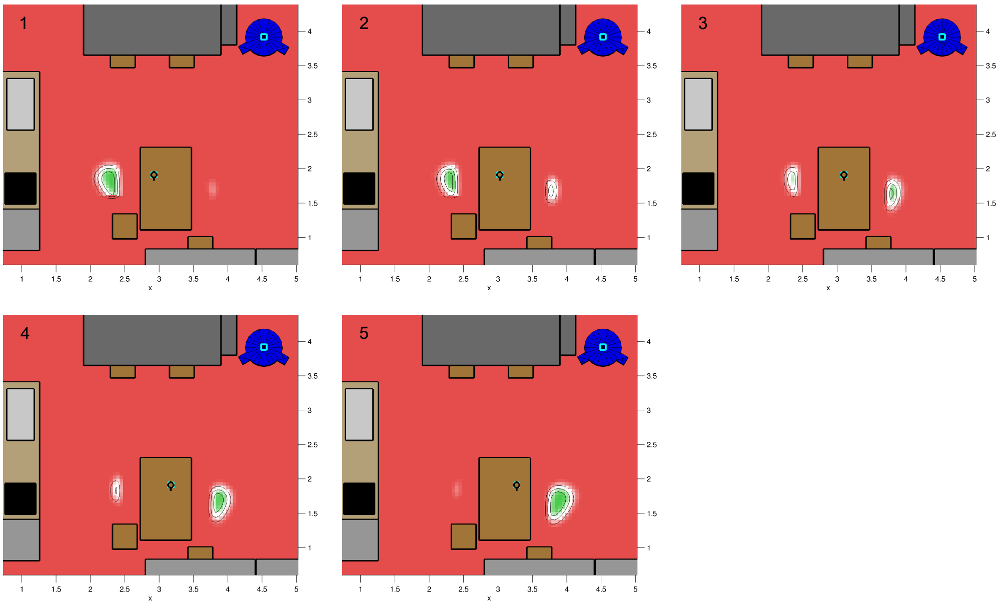


FIGURE 6.18 Plots of grasp success probability for all experiments. Each plot is labeled with the corresponding experiment number.

For task A), the average grasp success probability of the proposed manipulation place was 95% when using ARPLACE$_{\text{PROB}}$ compared to 77% when using ARPLACE$_{\text{UTIL}}$. The average travel time of the proposed manipulation place was 45.6s and 33.9s respectively. Therefore, on average, the robot using ARPLACE$_{\text{UTIL}}$ traded 18% of CGSP for 11.7s of travel time, which amounts to an increase in utility of 36.8%.

We also computed standard deviations for grasp success probability and estimated travel time. When performing task A) the standard deviation of grasp success probability is significantly higher when using ARPLACE$_{\text{UTIL}}$ with 0.28 compared to a standard deviation of 0.06 when using ARPLACE$_{\text{PROB}}$. This is intuitive because when time is the dominant factor, then ARPLACE$_{\text{UTIL}}$ will often choose grid cells with suboptimal grasp success probability. In fact, from a total of 441 experiments on performing task A), ARPLACE$_{\text{UTIL}}$ chose a grid cell with suboptimal grasp success probability in 153 cases.

When performing task B), this occurred in only 14 out of 441 experiments. The reason is that as the importance of success increases, the importance of saving execution time decreases relative to grasp success probability. This also explains why, when performing task B), average grasp success probability of the manipulation place that is proposed by ARPLACE$_{\text{UTIL}}$ approaches grasp success probability of the manipulation place that is proposed by ARPLACE$_{\text{PROB}}$ (0.955 compared to 0.956). Moreover, average estimated travel time of ARPLACE$_{\text{UTIL}}$ approaches estimated travel time of ARPLACE$_{\text{PROB}}$ with 45.1s and 46.2s respectively. When performing task B), ARPLACE$_{\text{UTIL}}$ proposes manipulation places that, on average, have 1%

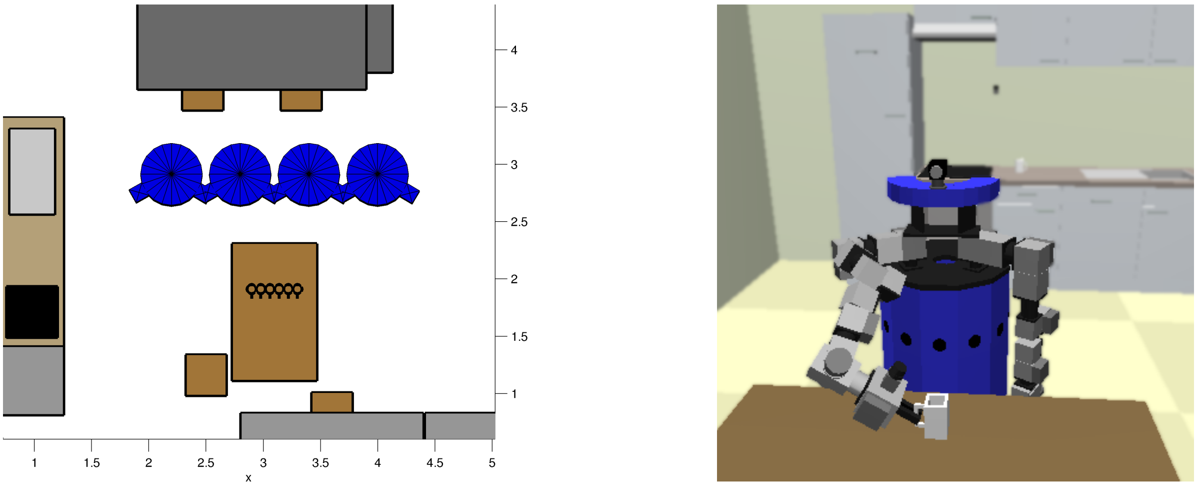


FIGURE 6.19 Left: Robot and object positions that are considered in the experiments for evaluating average behavior. Only four robot and six cup positions are shown for clarity. Right: Our B21r robot grasping a cup in the Gazebo simulator.

	Task A)	Task B)	Task C)	Overall
$\mu[p(x_{p^*}, y_{p^*})]$ / $\mu[p(x_{u^*}, y_{u^*})]$	0.95 / 0.77	0.96 / 0.95	0.96 / 0.96	0.95 / 0.89
$\sigma[p(x_{p^*}, y_{p^*})]$ / $\sigma[p(x_{u^*}, y_{u^*})]$	0.06 / 0.28	0.04 / 0.05	0.04 / 0.04	0.05 / 0.19
$\mu[t(x_{p^*}, y_{p^*})]$ / $\mu[t(x_{u^*}, y_{u^*})]$	45.6s / 33.9s	46.2s / 45.1s	46.2s / 46.2s	46.0s / 41.7s
$\sigma[t(x_{p^*}, y_{p^*})]$ / $\sigma[t(x_{u^*}, y_{u^*})]$	15.0s / 12.4s	14.6s / 14.7s	14.6s / 14.6s	14.7s / 15.0s

FIGURE 6.20 Analysis that compares the manipulation places that are proposed by ARPLACE$_{\text{PROB}}$ and ARPLACE$_{\text{UTIL}}$. Compared are mean grasp success probability ($\mu[p(x_{p^*}, y_{p^*})]$ / $\mu[p(x_{u^*}, y_{u^*})]$), mean travel time ($\mu[t(x_{p^*}, y_{p^*})]$ / $\mu[t(x_{u^*}, y_{u^*})]$), and the corresponding standard deviations $\sigma[\cdot]$ for tasks A) - C). The rightmost column aggregates the data for all three tasks. In each pair "A / B", A is the value for ARPLACE$_{\text{PROB}}$ and B the value for ARPLACE$_{\text{UTIL}}$.

less grasp success probability, but can be performed faster by 1.1s. Utility is increased by a mere 0.3%. When performing task C) the results are completely identical, because grasp success probability is clearly the dominant factor.

Averaging over the entire set of experiments, we observe that grasp success probability was 95% when using ARPLACE$_{\text{PROB}}$ and 89% when using ARPLACE$_{\text{UTIL}}$. Average travel time was 41.7s when using ARPLACE$_{\text{UTIL}}$ and 46.0s when using ARPLACE$_{\text{PROB}}$. Therefore, ARPLACE$_{\text{UTIL}}$ chose base positions that, on average, saved 9% of travel time but lead to a grasp success probability that was 6% below the optimum. In terms of performance, this implies that ARPLACE$_{\text{UTIL}}$ led to an expected 12.4% increase in utility.

With a further series of experiments, we show that this expected increase in performance is indeed observed when the model is applied to our B21r manipulation platform in simula-

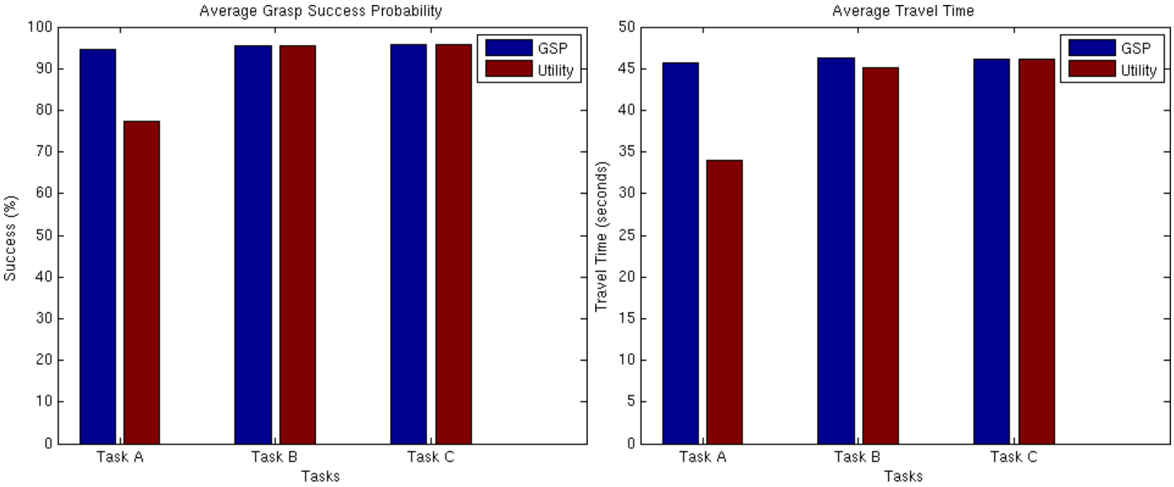


FIGURE 6.21 Left: Visualization of average grasp success probability of manipulation places that are proposed by ARPLACE$_{\text{PROB}}$ and ARPLACE$_{\text{UTIL}}$. This is a visualization of the data in the first row of Figure 6.20. Right: Visualization of average travel time (third row of Figure 6.20).

tion (see Figure 6.19 (right)). We considered 25 of the 441 experiment configurations named above (five initial robot positions and five cup positions) and applied, to each configuration, both ARPLACE$_{\text{PROB}}$ and ARPLACE$_{\text{UTIL}}$ six times for each of the tasks A)-C) (i.e. 900 experiments in total). In these experiments, the performance (i.e. the true utility) for tasks A)-C) was increased by 16.9%, 2.8% and 0% respectively (6.6% on average). Although we do not reach the predicted 12.4% increase, we observe that applying the more elaborate utility model leads to a performance increase whenever an increase is indeed possible to achieve.

The degree to which the observed utilities match the predicted utilities is determined exclusively by the accuracies of the underlying grasp success model and the navigation-time model. In our experiments, the navigation-time model described in section 6.2.3.1 had an average relative error of 18.3%. While this number may seem excessively large, we found that the motion controller exhibits a mean deviation of 14.4% in the actual execution times that are observed when it is applied a particular navigation problem. This is due to the controller's behaviour when overshooting waypoints (which occurs non-deterministically) and the corresponding recovery routines. Therefore, no navigation-time model can do significantly better. Indeed, a conceptually much simpler model that uses only distance to estimate navigation time results in an average relative error of 32.2% (even if the empirically optimal factor for the conversion from distance to time is used).

CHAPTER 7

Case Study

In this chapter we revisite the scenario that was introduced in chapter 2 and describe in detail how the ARPLACE framework tackles the questions that were raised. The goal of this chapter is twofold. First, the case study elaborates the evaluation of the last chapter. It is shown that utility-based ARPLACEs are robust and keep proposing promising base positions in complex scenarios. Second, the reader should get an intuition for how ARPLACEs behave and what manipulation places will be proposed in complex situations.

7.1 The Scenario

The task of the robot is to "clean the kitchen table" after its owners finished breakfast. The robot uses its knowledge base to infer that "cleaning the kitchen table" means to pick up all objects that are located on the table and put them into the dishwasher. The robot also knows the position of the dishwasher, because it remembers the dishwasher to be next to the stove and the fridge, and the position of all these objects remained static in the past.

The robot however, is not completely certain where the kitchen table is located. Although the robot has a good guess, it experienced that the position of the kitchen table changes several centimeters from time to time. Additionally, the robot does not know how many and which objects are located on the kitchen table. Therefore, the robot divides the task "clean the kitchen table" into a plan with several subtasks.

1. Find the kitchen table
2. Find target objects on the kitchen table
3. Pick up target object and put it into the dishwasher

The first subtask has to be achieved before the second and third subtask. The second subtask has to be achieved before the third subtask. When at least one target object is detected, the second and third subtask can be executed interleaved. In case there are multiple target objects, the robot has to perform the third subtask multiple times. The base positions from where the robot performs manipulation actions are computed by the ARPLACE framework. If appropriate, the ARPLACE framework can propose base positions from where two objects can be grasped at once.

The left image of Figure 7.1 depicts an overview of the scenario before the robot enters the kitchen. We see that there are two target objects on the kitchen table: a glass and a cup. Another cup is positioned between the stove and the sink. Because it is not located on the kitchen table, it is no target object in the current task. The right image of Figure 7.1 shows the robot's internal representation of the scenario according to its current knowledge. Only the static objects are in its map.

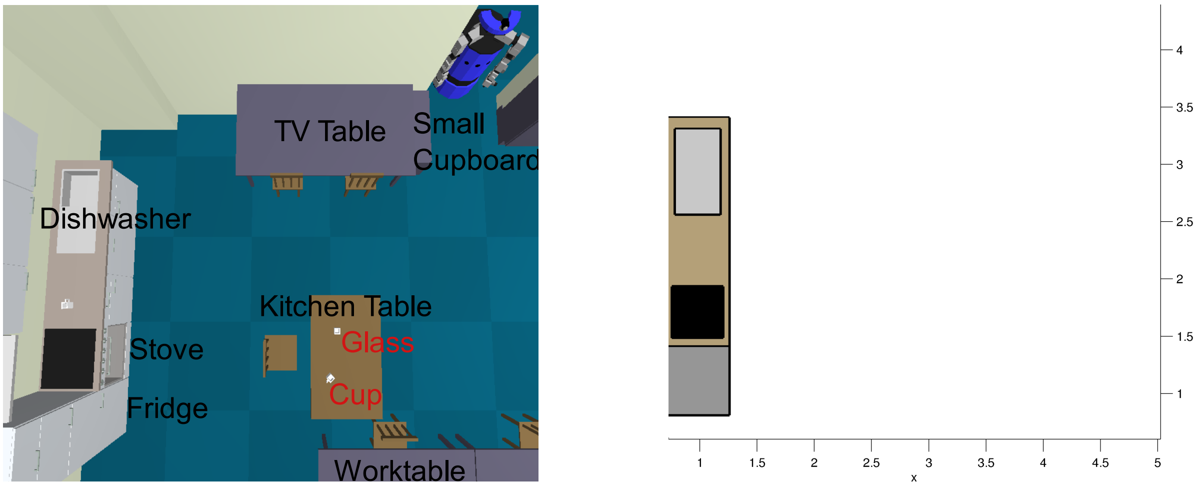


FIGURE 7.1 Left: Scenario before robot enters the kitchen. Right: The robot's internal view on the world according to its current knowledge.

7.2 Finding Target Objects

As the robot enters the kitchen, new sensor data arrives. In the left image of Figure 7.2 the robot discovered several objects by analyzing its laser range data. The discovered objects are the TV table on the top of the image, the small cupboard beneath it, the two worktables on the bottom of the image, and a chair at the left worktable. The bright blue square in the robot represents the robot's uncertainty into its current base position, as computed by its particle filter based Adaptive Monte Carlo Localization algorithm. The robot's current positional uncertainty is $\sigma_{x_{\text{rob}}} = \sigma_{y_{\text{rob}}} = 0.04\text{m}$. The robot also maintains corresponding uncertainties for object positions, but they are not depicted.

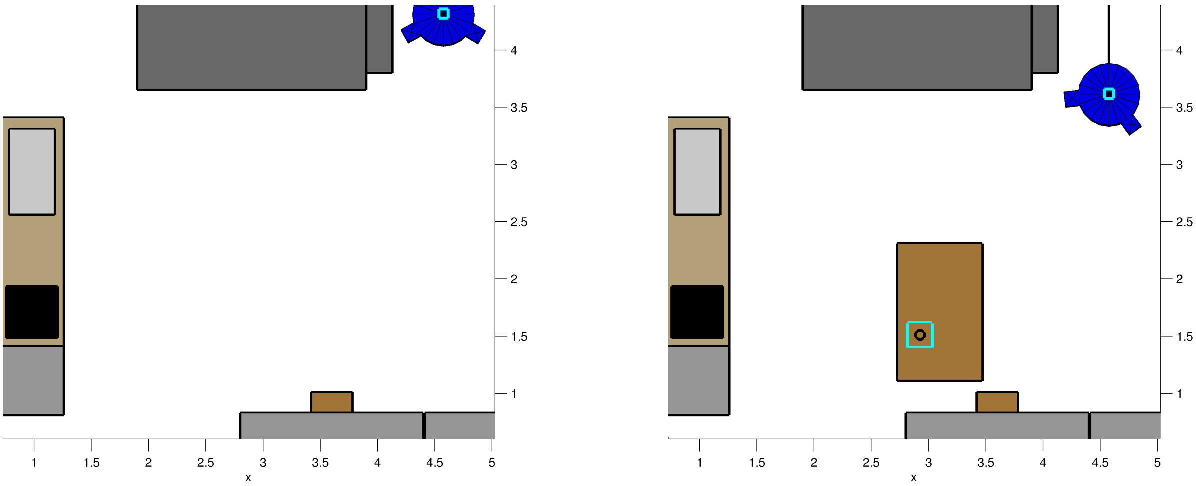


FIGURE 7.2 Left: Robot enters kitchen and discovers several objects by analyzing its laser range data. The bright blue square in the robot visualizes its localization uncertainty. Right: The robot discovered the kitchen table and a target object. The bright blue square around obj_1 is the robot's uncertainty about the object's pose.

The robot continues to move into the kitchen and finds the kitchen table. Plan step 1) "Find the kitchen table" is achieved. The next plan step is "Find target objects on the kitchen table". Therefore, the robot starts to turn towards the kitchen table. While turning, the robot's vision system detects an object (obj_1) on the table, as can be seen in the right image of Figure 7.2. The estimated pose of the object is $obj = \langle 2.93\text{m}, 1.50\text{m}, \text{undef.}\rangle$. The robot also tries to estimate the type of the target object. Therefore, the vision system matches the shape of obj_1 against an internal database of 3D CAD models and considers obj_1 to be a glass or a cup. Because no handle is detected there is a slight preference for a glass. The estimation about a target object's type is stored in a probability vector typ_{obj}. For n different objects $\{o_1, .., o_n\}$ that can be recognized by the vision system, $typ_{obj} = \{p_1, .., p_n\}$ stores the probabilities p_i that obj is of a certain object type, where $i \in \{o_1, .., o_n\}$. In our case $typ_{obj_1} = \{p_{cup}, p_{glass}\}$ with two elements that store the probability that the target object is a cup (p_{cup}) or a glass (p_{glass}). As no handle is detected, the vision system has a slight preference for a glass and the probability distribution is $typ_{obj_1} = \langle 0.45, 0.55\rangle$. Moreover, the lack of a handle impedes the robot from computing the object's angle on the table. This is why the angle in the pose of obj_1 is estimated as being "undef.". The uncertainty into the pose of obj_1 is high ($\sigma_{x_{\text{obj}_1}} = \sigma_{y_{\text{obj}_1}} = 0.12\text{m}$) because the object is small, far away, and little sensor data is available yet.

7.3 Computing Manipulation Places

Plan step 2) "Find target objects on the kitchen table" is achieved and the robot now executes plan steps 2) and 3) interleaved. The primary goal now is to "Pick up target objects and put them into the dishwasher", but if new target objects are detected then they will be taken into account immediately. In order to find a promising manipulation place for grasping obj_1, the robot queries the ARPLACE framework. This includes five steps: (1) use Generalized Success Models for computing grasp success probability; (2) compute the probability that positions are blocked by obstacles; (3) merge grasp success probability and obstacle probability into a single probability value; (4) estimate travel time; (5) merge grasp success probability and estimated travel time to a utility value. The center of the grid cell with highest utility will be the proposed manipulation place and chosen as the robot's goal position for navigation.

Figure 7.20 at the end of the chapter presents an overview of all computational steps and intermediate representations that are computed.

7.3.1 Grasp Success Probability

The rest of this section describes how to compute the ARPLACE under the condition that the target object's type is uncertain. Because the robot is not completely certain whether obj_1 is a cup or a glass, grasps from the side for grasping a cup at its handle and from the top for grasping a glass at its body have to be considered. Therefore the robot has the following options to grasp obj_1

- Grasp obj_1 from the top with the right arm
- Grasp obj_1 from the side with the right arm
- Grasp obj_1 from the top with the left arm
- Grasp obj_1 from the side with the left arm

7.3.1.1 Grasping with the Right Arm

We start by considering grasps with the right arm. Both plots in Figure 7.3 show the ARPLACE probability distribution for grasping obj_1 from the top with the right arm. It can be seen that there are three clusters with grasp success probabilities that are higher than 0%. Every cluster corresponds to a table edge from where the manipulation action can be performed. There is no cluster for te_4, because obj_1 is out of reach from there. The probability to grasp obj_1 from te_3 is low. The reason is that although within reach, the object is far away which would require the robot to significantly stretch its arm, which leads to a difficult reaching motion. Even worse, if the robot's estimation about its base position is slightly off, then obj_1 could be out of reach.

Grasp success probability from te_2 is higher, because obj_1 is nearer. When grasping a glass from the top, then the angular orientation of the glass can be neglected because there are no constraints on the final yaw-orientation of the gripper. There are two black isobars drawn within $cluster_{te_2}$. Isobars are drawn at grasp success probability levels of 20%, 50%, and 80%. Therefore, the maximal grasp success probability for grasping obj_1 from te_2 has a probability of more than 50% but less than 80%. Please note that all furniture is drawn at a height of 0.5 and so the probability distribution of grid cells with lower grasp success probability than 50% is hidden.

Grasp success probability from te_1 includes the grid cell with the highest probability of success p^*_{rt}. The index rt means that in this case we consider grasps with the *r*ight arm from the *t*op. In the left image of Figure 7.3 $p^*_{rt} = 81.3\%$ and the center of the corresponding grid cell is located at $c(x_{p^*_{rt}}, y_{p^*_{rt}}) = \langle 2.35\text{m}, 1.65\text{m}\rangle$. The reason why grasp success probability does not approach 100% is that the robot's uncertainty into the object's pose is high and there is medium uncertainty into its base position.

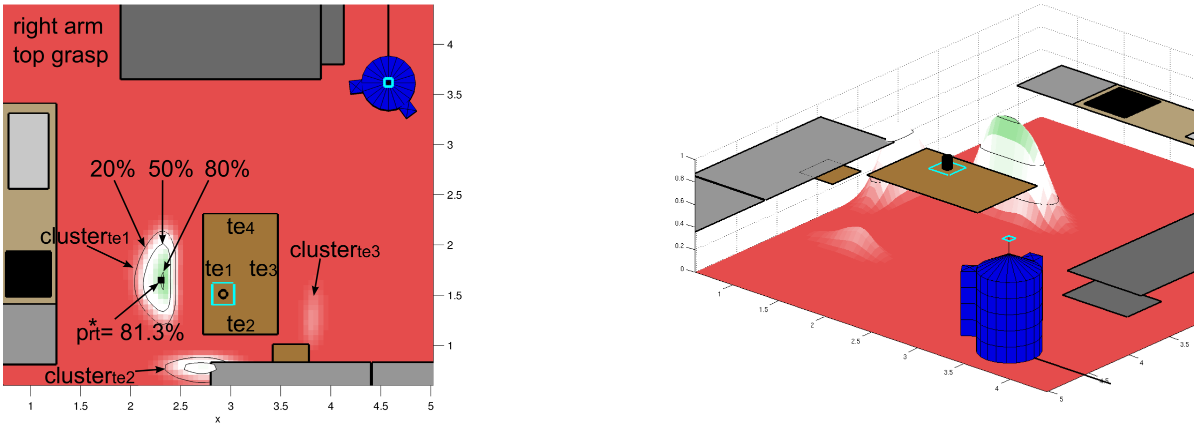


FIGURE 7.3 Left: ARPLACE probability distribution for grasping obj_1 from the top with the right arm. The black lines are isobars that visualize areas with a grasp success probability of more than 20%, 50%, and 80%. The black square marks (x_{p^*}, y_{p^*}) which is the grid cell with the highest grasp success probability. Right: Same plot from an isometric perspective.

There is the possibility that the handle of obj_1 is hidden by its body, and obj_1 is a cup instead of a glass. This possibility is reflected by the robot's believe state about the type of obj_1 which was found to be $typ_{obj_1} = \{0.45, 0.55\}$, so the robot expects obj_1 to be a cup with a probability of 45%. If obj_1 is a cup, the robot wants to grasp it from the side at its handle. The ARPLACE probability distribution for grasping obj_1 from the side with the right arm is depicted in the center image of Figure 7.4. It can be seen that grasping obj_1 from the side is possible only from the cluster at te_1. $p^*_{rs} = 80.8\%$ at the grid cell that is centered at $\langle 2.20\text{m}, 1.50\text{m}\rangle$. The index rs means that in this case we consider grasps with the right arm from the side.

The next step is to compute the ARPLACE probability distribution for grasping obj_1 with the right arm without considering the type of grasp. This is done by merging the ARPLACE probability distributions for grasping obj_1 from the top and from the side according to the robot's believe state of how probable the corresponding grasp will be. The merging of ARPLACEs is depicted in Figure 7.4, where the left plot corresponds to grasping obj_1 from the top and the center plot corresponds to grasping obj_1 from the side. The right plot visualizes the resulting ARPLACE probability distribution. It is the weighted sum of the two source plots. Although both source plots have manipulation places with a grasp success probability above 80%, the merged plot lacks such a region. The reason is that the most promising manipulation places of the two source plots do not overlap. This can be seen in the right image of Figure 7.4, where the maximal values for all ARPLACE probability distributions are marked black. For the resulting plot, $p^*_r = 66.2\%$ and is located at the grid cell that is centered at $c(x_{p^*_r}, y_{p^*_r}) = \langle 2.30\text{m}, 1.50\text{m}\rangle$, which is roughly between the maximum probabilities for top and side grasps. The index r means that we consider grasps with the right arm and any approach direction.

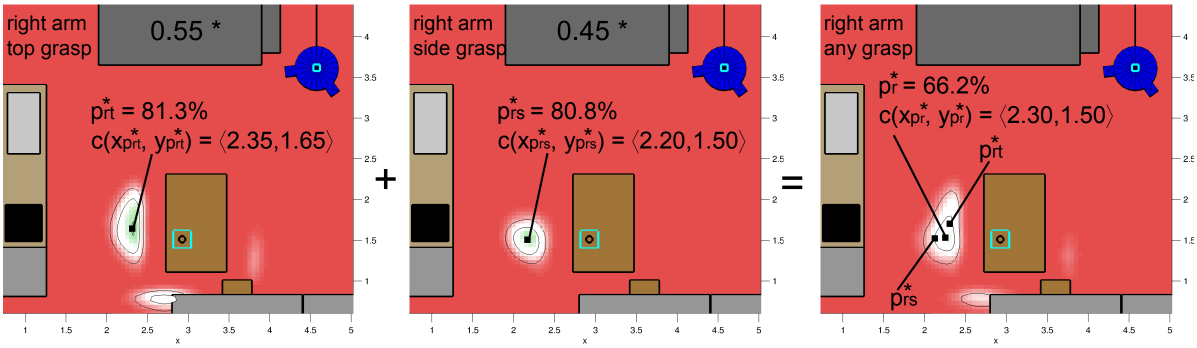


FIGURE 7.4 Left: ARPLACE probability distribution for grasping obj_1 with the right arm from the top (left image) and from the side (center image). Right: Merged ARPLACE probability distribution for grasping obj_1 with the right arm when considering that the robot expects that obj_1 has to be grasped from the top with a probability of 55% and from the side with a probability of 45%. Grid cells with the highest grasp success probability are marked in each plot.

7.3.1.2 Grasping with the Left Arm

The same computations are performed for grasping obj_1 with the left arm. The results are depicted in Figure 7.5.The maximal probability for grasping obj_1 from the top with the left arm is $p^*_{lt} = 84.6\%$ at the grid cell that is centered at $\langle 2.35\mathrm{m}, 1.40\mathrm{m}\rangle$, as can be seen in the left image of Figure 7.5. The maximal probability for grasping obj_1 from the side with the left arm is $p^*_{ls} = 41.4\%$ at the grid cell that is centered at $\langle 2.75\mathrm{m}, 0.60\mathrm{m}\rangle$, as can be seen in the center image of Figure 7.5.

The right plot in Figure 7.5 depicts the ARPLACE probability distribution for grasping obj_1 with the left arm and any type of grasp. It is obtained by merging the left and center plot. $p^*_l = 46.5\%$ at the grid cell that is centered at $c(x_{p^*_l}, y_{p^*_l}) = \langle 2.35\mathrm{m}, 1.40\mathrm{m}\rangle$, which is the same grid cell as p^*_{lt}.

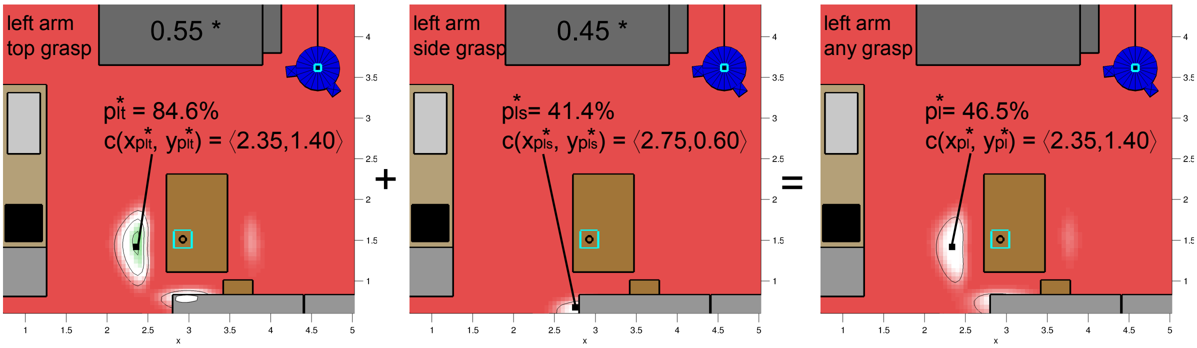


FIGURE 7.5 ARPLACE probability distribution for grasping with the left arm from the top (left image) and with the left arm from the side (center image). Right: Merged ARPLACE probability distribution for grasping with the left arm.

Because subsequent processing steps make use of the grasp success probability value without considering if it was derived from the right or left arm we can safely merge the ARPLACEs for grasping with the right arm and the and the ARPLACE for grasping with the left arm into a single ARPLACE. This enables us to save computation timein further computations because they have to be performed only on the merged ARPLACE.

7.3.1.3 Merging ARPLACEs for Left and Right Arm

Merging is done by using the max-operator as shown in Figure 7.6. It can be seen that the area that is not completely red grows, meaning that the number of grid cells with a grasp success probability of more than 0% increases. When analyzing the merged ARPLACE probability distribution, it can be seen that the cluster at te_1 is dominated by the cluster of the right arm, while the cluster of the left arm stretches it out to the bottom and a little bit to the right. The grid cell with maximum grasp success probability is centered at position $c(x_{p^*}, y_{p^*}) = \langle 2.35\mathrm{m}, 1.50\mathrm{m}\rangle$, has a grasp success probability of $p^* = 66.2\%$, and proposes to use the right arm for grasping. Overall it is intuitive that the ARPLACE framework proposes to perform the manipulation action with the right arm from $\langle 2.35\mathrm{m}, 1.50\mathrm{m}\rangle$. The reason is that the robot is not certain about the type of the target object. It could be a glass or a cup. The proposed base position reaches high grasp success probabilities for both types of objects.

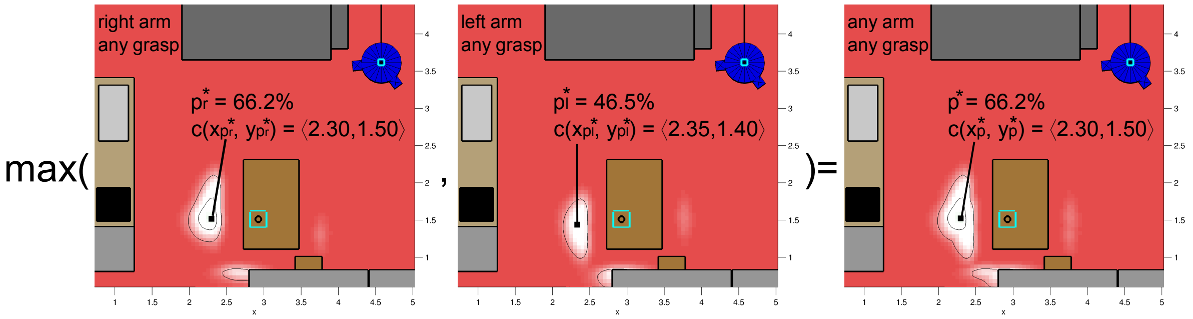


FIGURE 7.6 ARPLACE probability distribution for grasping obj_1 with right arm (left image) and with left arm (center image). Right: Merged probability distribution for grasping with any arm. For every grid cell, the merged grasp success probability is the maximal probability of successfully grasping with the right or left arm.

7.3.2 Considering Obstacles

Grasp success probability is based on the estimated pose of the target object and the state estimation uncertainties that the robot has into its base position and the target object's pose. Another important issue is to take obstacles into account which may block promising base positions. The robot estimates an obstacle in 2D space by its mean pose, length, width, and angular orientation, as well as a covariance matrix that represents the robot's uncertainty into the estimated obstacle pose. The center image of Figure 7.7 visualizes an obstacle probability distribution for the current situation. Please note that obstacles are grown by the robot's radius. While the robot is absolutely certain about the pose of the fridge, cooker and sink, it has a high uncertainty into the table's pose ($\sigma_{x_{tab}} = \sigma_{y_{tab}} = 0.06\text{m}$), because the robot just discovered it and little sensor data is available. The uncertainty into the other objects is relatively low with standard deviations around 0.02m for the x-axis and y-axis.

Because the obstacle probability distribution is discretized into the same grid cells as the ARPLACE probability distribution, both probability distributions can be merged easily. For every grid cell, its (basic) grasp success probability is multiplied by the probability that it is unoccupied. The resulting probability value is the new (combined) grasp success probability. After this computational step, the cluster for te_3 got smaller at the bottom as can be seen in the right image of Figure 7.7. The reason is that this area of promising manipulation places is blocked by the chair at worktable 1. Even grid cells with high grasp success probability are cancelled out, if the grid cell is known to be occupied. The cluster at te_2 gets much smaller at the right side, because it is blocked by worktable 1. The kitchen table makes the cluster at te_2 a little bit smaller at the top. Furthermore, the kitchen table reduces the size of the cluster at te_1 at the right side.

The obstacle probability distribution also affects grid cells inside the ARPLACE probability distribution. Maximum (basic) grasp success probability for grasping with the left arm, for example, was $p_l^* = 0.465$ at the grid cell that is centered at $\langle 2.35\text{m}, 1.40\text{m}\rangle$. This was depicted in Figure 7.5. The probability that this grid cell is unoccupied is $p_U(x_{p_l^*}, y_{p_l^*}) = 0.968$, and combined grasp success probability therefore is $0.465 * 0.968 = 0.450$. The distance of $(x_{p_r^*}, y_{p_r^*})$ that is centered at $\langle 2.30\text{m}, 1.50\text{m}\rangle$ to the table is bigger by 5cm, and because $p_U(x_{p_r^*}, y_{p_r^*}) = 1.0$ at this grid cell combined grasp success probability remains at 66.2%.

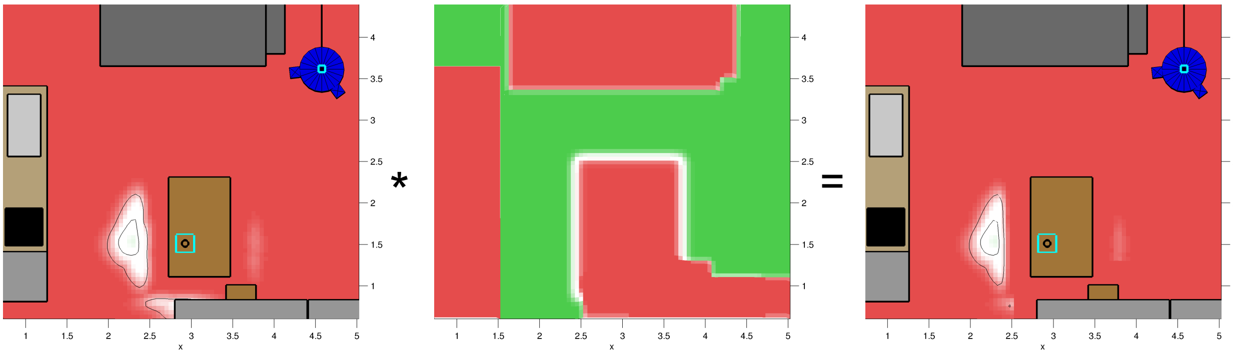


FIGURE 7.7 Left: ARPLACE probability distribution based on basic grasp success probability. Center: Obstacle probability distribution. Green grid cells are predicted to be completely free, so the probability of being unoccupied is 100%. White and red grid cells are predicted to be unoccupied with successively lower probability. Right: ARPLACE probability distribution based on combined grasp success probability.

7.3.3 Computing Grasp Utility

If the robot's only concern is to grasp target objects as robustly as possible, then manipulation actions should be performed from within the grid cell with the highest combined grasp success probability. However, sometimes robust grasping is not the robot's only concern. There may be additional constraints like execution time or power consumption that have to be taken into account. Therefore, we want to optimize the following utility heuristic

$$u(x, y) = p(x, y) \cdot w_S + u_T(x, y) \cdot w_T$$

where $p(x, y)$ is combined grasp success probability at grid cell (x, y), w_S is importance of success, $u_T(x, y)$ is the utility of time at grid cell (x, y), and w_T is the weight for weighting the utility of time. w_S and w_T are computed by a high-level planner according to the task context. Higher values for w_T represent that it is urgent to perform the task as quickly as possible, and higher values for w_S represent that it is important to successfully perform the grasping task in order to avoid undesired drawbacks such as spilling juice over the ground, or breaking the object. In this scenario the task is not urgent because the owners do not need the kitchen in the near future and no other tasks are scheduled for the robot. That is why w_T is set to 0.1 by the high-level planner. w_S is set to 30 seconds because the target object is empty and not particularly valuable.

The center image of Figure 7.8 depicts the result of estimating travel time from the robot's current base position to every grid cell. The grid cell that can be reached fastest and has a grasp success probability of more than 0% is abbreviated (x_t^*, y_t^*) and is centered at $\langle 3.85\text{m}, 2.10\text{m}\rangle$ with an estimated travel time of $t^* = 40.1\text{s}$. The travel time to the grid cell with maximal utility of success is estimated to be 66.3s.

The right image of Figure 7.8 depicts the resulting utility distribution. Compared to the left image where grasp success probability is plotted, the area of promising grid cells seems to have grown. Several grid cells that were completely red in the left plot changed their color to a brighter red. This is an effect of our filter that sets the utility value of every grid cell with a grasp success probability of 0% to $-\infty$s. Therefore, grid cells that have a grasp success probability of slightly over 0% are almost completely red in the left plot, while their utility value in the right plot is low but high enough to result in a brigther red than the grid cells that have a utility value of $-\infty$s.

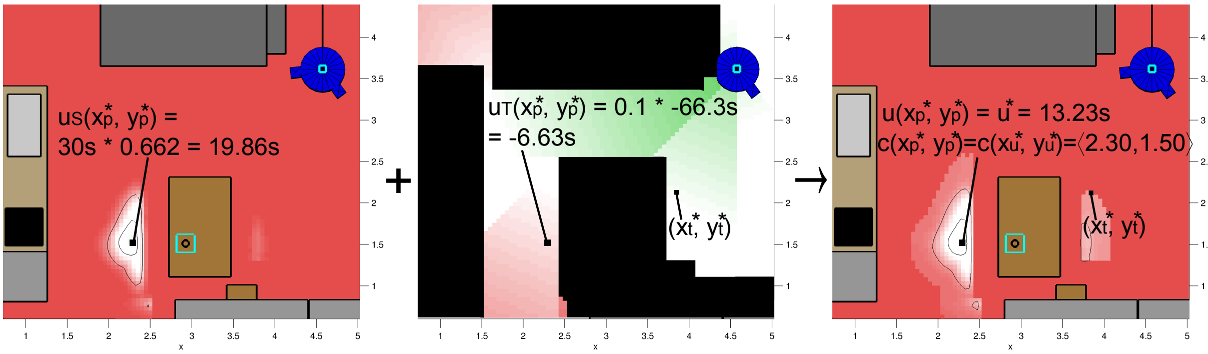


FIGURE 7.8 Left: ARPLACE probability distribution. Center: Estimated travel time from the robot's current position to every grid cell. Green indicates low travel time, white and red successively longer times. Unreachable cells are black. Right: Merged ARPLACE utility distribution.

In this example, the grid cell with maximum utility value (u^*) is identical to the grid cell with the highest grasp success probability. The corresponding grid cell (x_{u^*}, y_{u^*}) is centered at $\langle 2.30\text{m}, 1.50\text{m}\rangle$, has a grasp success probability of 66.2% and an estimated travel time of 66.3 seconds. Its utility value is computed as follows

$$u^* = 30\text{s} \cdot 0.662 + 0.1 \cdot (-66.3\text{s}) = 13.23\text{s}$$

The center of the grid cell with the highest utility $c(x_{u^*}, y_{u^*})$ is transferred from the ARPLACE framework to the navigation system as the proposed manipulation place. It is furthermore returned to the transformational planning system and the high-level planner.

Overall, it is intuitive that ARPLACE chose $\langle 2.30\text{m}, 1.50\text{m}\rangle$ as manipulation place. First, the robot is certain that the grid cell is not blocked by an obstacle ($p_U(x_{u^*}, y_{u^*}) = 1.0$). Second, the robot is not certain about the type of the target object. It could be a glass or a cup. The proposed base position reaches high grasp success probabilities for both types of objects. The only drawback is that the estimated travel time is higher than for any grid cell in the cluster of te_3. But since the difference is not striking (66.3s compared to at least 40.1s), and reducing execution time is only a minor goal in the current task, the robot should prefer base positions with high grasp success probability.

7.4 Updating ARPLACEs

After having located a target object and finding the most promising manipulation place, the robot moves towards the manipulation place. While moving, new sensor data arrives. The ARPLACE framework takes the new data into account as can be seen in Figure 7.9. The left image shows the robot at base position $\langle 4.2\text{m}, 3.2\text{m}, \frac{5}{4}\pi\rangle$. The updated versions of the ARPLACE probability and utility distributions are depicted in the center and right image.

Due to additional laser range information, the positional uncertainties of all furniture objects decreased. For example, the standard deviation of the kitchen table decreased from $\sigma_{x_{obj_1}} = \sigma_{y_{obj_1}} = 0.06\text{m}$ to $\sigma_{x_{obj_1}} = \sigma_{y_{obj_1}} = 0.02\text{m}$. And the standard deviation of the target object's pose decreased from $\sigma_{x_{tab}} = \sigma_{y_{tab}} = 0.12\text{m}$ to $\sigma_{x_{tab}} = \sigma_{y_{tab}} = 0.05\text{m}$. Reduced uncertainty leads to higher grasp success probability. The number of grid cells with high grasp success probability increased as can be seen when comparing the isobars in the center image of Figure 7.9 and the right image of Figure 7.7. There are several grid cells with a grasp success probability of 80% and more in Figure 7.9. p^* changed from 66.2% in Figure 7.7 to 89.6% in Figure 7.9. The proposed manipulation place changed from $\langle 2.30\text{m}, 1.50\text{m}\rangle$ to $\langle 2.30\text{m}, 1.55\text{m}\rangle$.

Another fact worth mentioning is that the robot did not detect a handle yet, and the probability of obj_1 being a glass raised from $typ_{obj_1} = \langle 0.45, 0.55\rangle$ to $typ_{obj_1} = \langle 0.3, 0.7\rangle$. As a result, the grasp success probability cluster at te_1 got more symmetric. This can be seen when observing the bottom left part of the ARPLACE cluster at te_1 of Figure 7.7 and Figure 7.9. The utility value changed from 13.23s in Figure 7.7 to 20.77s in Figure 7.9. The higher utility value can be attributed mostly to higher grasp success probability, but the estimated travel time got shorter as well.

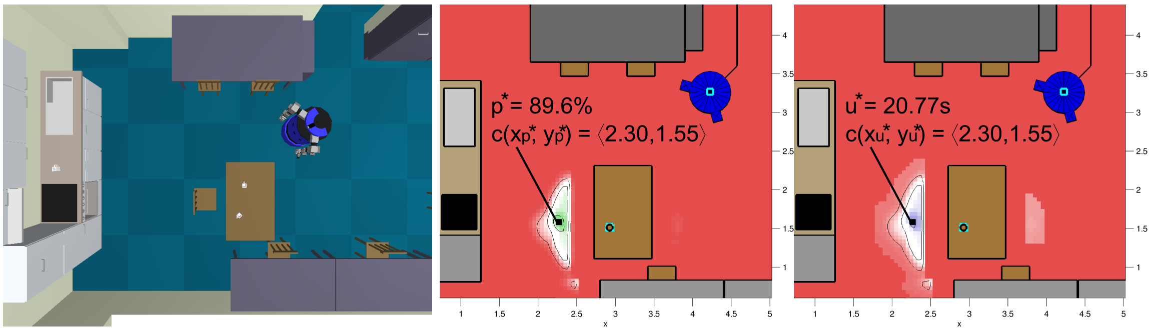


FIGURE 7.9 ARPLACEs get updated as new sensor data arrives.

To sum up, new sensor data allowed the robot to reduce its state estimation uncertainties, without detecting major flaws in the previous state estimation. As a result grasp success probability of the most promising base position increased considerably from 66.2% to 89.6%. The most promising manipulation place itself remained nearly the same by changing from $\langle 2.30\text{m}, 1.50\text{m}\rangle$ to $\langle 2.30\text{m}, 1.55\text{m}\rangle$. This is the new navigation goal that ARPLACE reports to the navigation system.

7.5 Handling Multiple Objects

The robot keeps moving forward and detects a second object on the kitchen table, as can be seen in the left image Figure 7.10. The robot estimates the pose of obj_2 to be $\langle 3.0\text{m}, 1.9\text{m}, \text{undef.}\rangle$ with a positional uncertainty of $\sigma_{x_{obj2}} = \sigma_{y_{obj2}} = 0.08\text{m}$. The object's type is estimated as $typ_{obj_2} = \langle 0.25, 0.75\rangle$. The second target object introduces an additional subtask and the robot's plan now consists of the subtasks

- "put obj_1 into the dishwasher"
- "put obj_2 into the dishwasher"

that can be accomplished in any order.

Figure 7.11 depicts the robot's preferred base positions for grasping obj_1 and obj_2. $u^*_{obj_1}$ is in the grid cell that is centered at $\langle 2.30\text{m}, 1.55\text{m}\rangle$ and proposes to use the right arm for grasping.

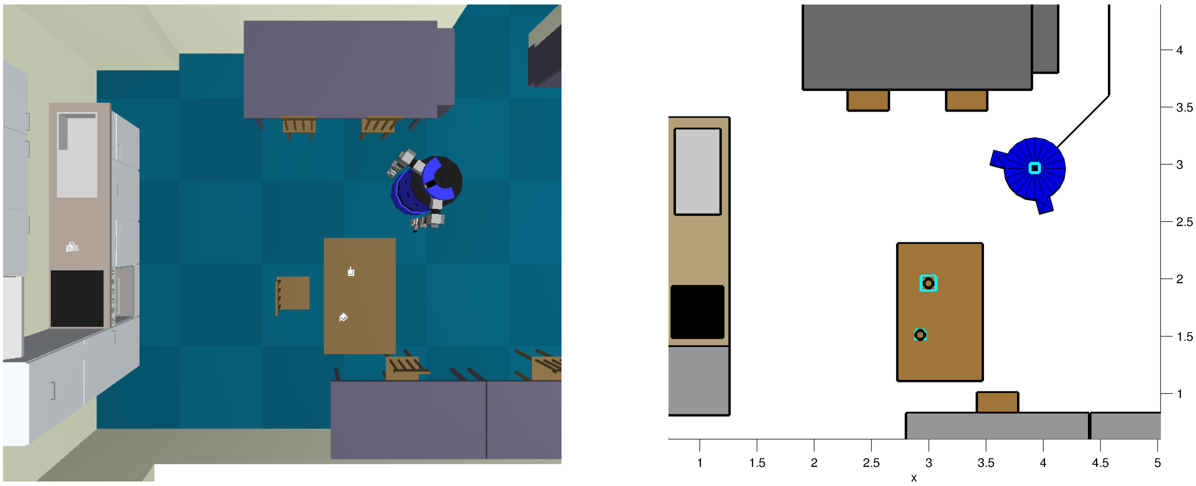


FIGURE 7.10 Left: Scenario when robot detects a second target object. Right: The robot's internal view on the world according to its current knowledge.

Grasp success probability is 90.0% and the estimated travel time is 56.57s, which leads to a utility value of 21.3s. $u^*_{obj_2}$ is in the grid cell that is centered at $\langle 2.35\text{m}, 2.05\text{m}\rangle$ and proposes to use the right arm for grasping. Grasp success probability is 86.1% and the estimated travel time is 51.36s, which leads to a utility value of 20.7s.

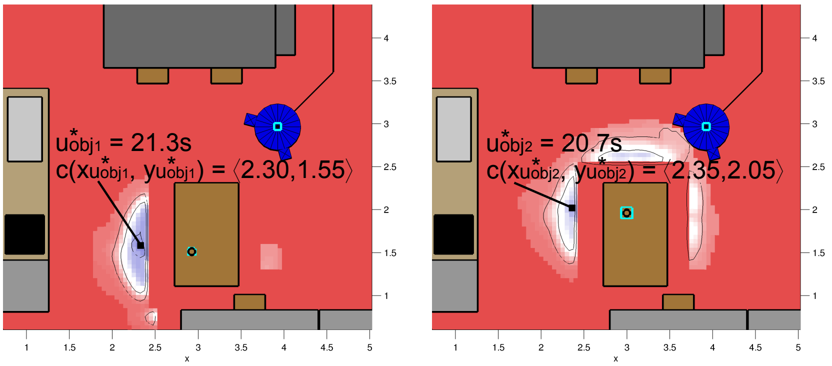


FIGURE 7.11 ARPLACE utility distributions for grasping obj_1 (left image) and obj_2 (right image).

However, the manipulation of multiple objects provides opportunities to further optimize base positioning, because the subgoals can be achieved with different action sequences.

A) Grasp obj_1 | Put obj_1 into dishwasher | Grasp obj_2 | Put obj_2 into dishwasher
B) Grasp obj_2 | Put obj_2 into dishwasher | Grasp obj_1 | Put obj_1 into dishwasher
C) Grasp obj_1 | Grasp obj_2 with other arm | Put obj_1 and obj_2 into dishwasher
D) Grasp obj_2 | Grasp obj_1 with other arm | Put obj_1 and obj_2 into dishwasher
E) Grasp obj_1 and obj_2 from same position | Put obj_1 and obj_2 into dishwasher

Because the ARPLACE framework only considers the next grasping action, plan A) and C) leads to the utility value of grasping obj_1, while plan B) and D) leads to the utility of grasping obj_2. The following section will show, how the utility value of plan E) is computed, which requires to compute the utility value $u^*_{obj_{12}}$ of grasping both objects at once from the same base position.

7.5.1 Merging ARPLACEs based on Utility

Grasping both objects at once can be achieved by the following possibilities.

1. Grasp obj_1 with the right arm and obj_2 with the left arm (*RL* grasp scheme)
2. Grasp obj_1 with the left arm and obj_2 with the right arm (*LR* grasp scheme)

Grasp success probability for successfully grasping both objects from the same base position is the product of the ARPLACE probability distribution for grasping obj_1 and the ARPLACE probability distribution for grasping obj_2. When considering the *RL* grasp scheme, then grasp success probability for successfully grasping both objects at once is the product of successfully grasping obj_1 with the right arm and grasping obj_2 with the left arm. We call the resulting ARPLACE probability distribution p_{rl}. p_{lr} is the ARPLACE probability distribution of using the *LR* grasp scheme. for grasping both objects at once. p_{lr} is computed as the product of successfully grasping obj_1 with the left arm and grasping obj_2 with the right arm. Figure 7.12 depicts the computation of p_{rl} (images on top) and p_{lr} (images at bottom).

In our scenario $p^*_{rl} = 60.9\%$ at the grid cell that is cenetered at $\langle 2.35\text{m}, 1.80\text{m}\rangle$. $p^*_{lr} = 17.9\%$ at the grid cell that is cenetered at $\langle 2.40\text{m}, 1.75\text{m}\rangle$. This means that grasp scheme *RL* is preferred. This is intuitive, as grasp scheme *LR* would require the robot to reach far to the left with the right arm, and far to the right with the left arm. In order to compute the utility of grasping obj_1 and obj_2 from the same base position, we merge p_{rl} and p_{lr} with the max-operator, and combine it with the utility of time. Figure 7.13 depicts the resulting ARPLACE utility distribution. $u^*_{obj_{12}}$ is at the grid cell that is centered at $\langle 2.35\text{m}, 1.80\text{m}\rangle$ and proposes to grasp obj_1 with the right arm and obj_2 with the left arm. Maximum grasp success probability for successfully grasping both at once objects is 60.9% and the estimated travel time is 53.9s, which leads to a utility value of 12.9s.

$u^*_{obj_{12}}$ is considerably lower than the utilities of $u^*_{obj_1}$ which was 21.3s, and $u^*_{obj_2}$ which was 20.7s. The default approach would lead to plan A) or plan C), because when the high level planner asks for a suitable base position for grasping, then ARPLACE will return the grid cell with maximal utility $c(x^*_{obj_1}, y^*_{obj_1})$.

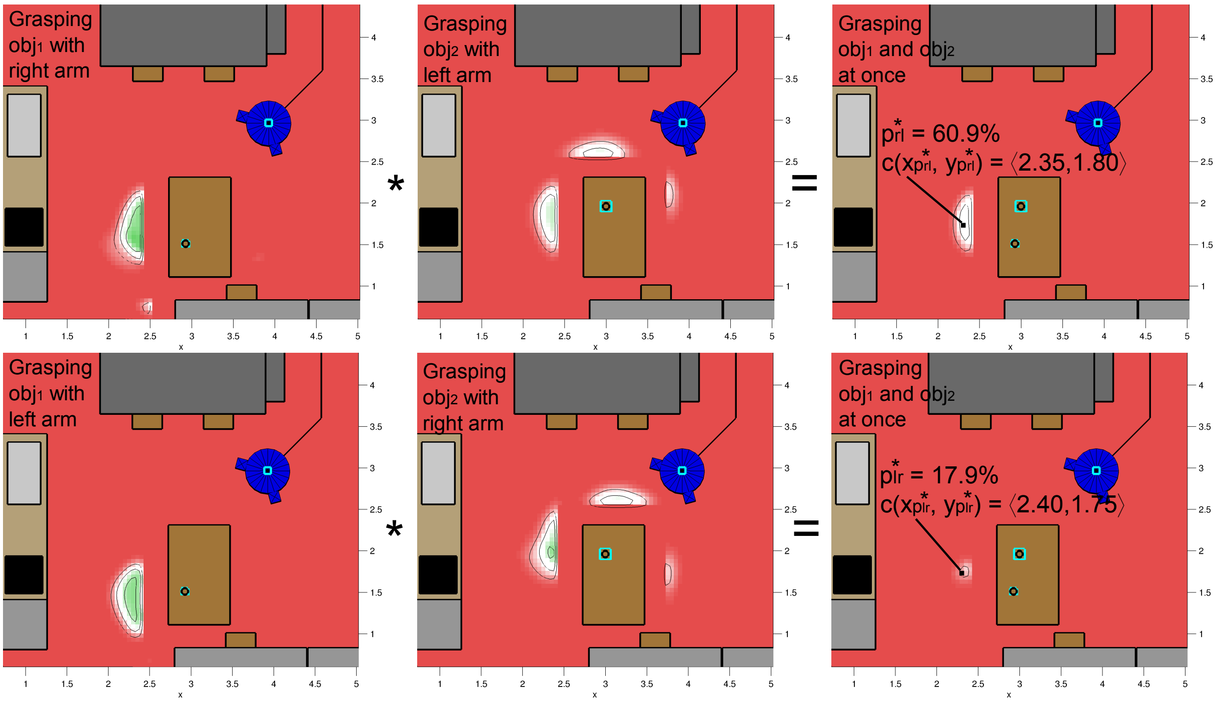


FIGURE 7.12 Top: Grasp success probability for grasping obj_1 with the right arm (left image), grasp success probability for grasping obj_2 with the left arm (center image), and grasp success probability for grasping both objects at once (right image). Bottom: The same for grasping both object at once and using the left arm for obj_1 and the right arm for obj_2.

However, the utility heuristic presented above is not fair to the plan of grasping both objects at once. It is obvious that plans E) might save overall execution time. The current utility heuristic accounts only for the next action and does not consider future actions. Therefore, the utility of moving to the best manipulation place for grasping a single object will always be higher than the utility for grasping multiple objects at once. The reason is that finding the most promising base position for grasping multiple objects means to find a base position that is a good compromise between the probabilities of successfully grasping obj_1 and obj_2. Moving to a base position that is better suited to grasp the one object will usually lead to a base position that is worse for grasping the other object. The only exception where $u^*_{obj_{12}}$ is equal to $max(u^*_{obj_1}, u^*_{obj_2})$ is when the condition $(x_{u^*_{obj_1}}, y_{u^*_{obj_1}}) = (x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$ holds. Or less formal: The grid cell with maximal utility for grasping obj_1 is identical to the grid cell with maximal utility for grasping obj_2

There are two possibilities to make the decision whether the robot should grasp both objects from a single base position. One opportunity is to use a transformational planning system, as described in 5.6.3.2. The other possibility is to compute overall utility of plans and choose the one that maximizes it. In this chapter we will focus on the utility-based approach.

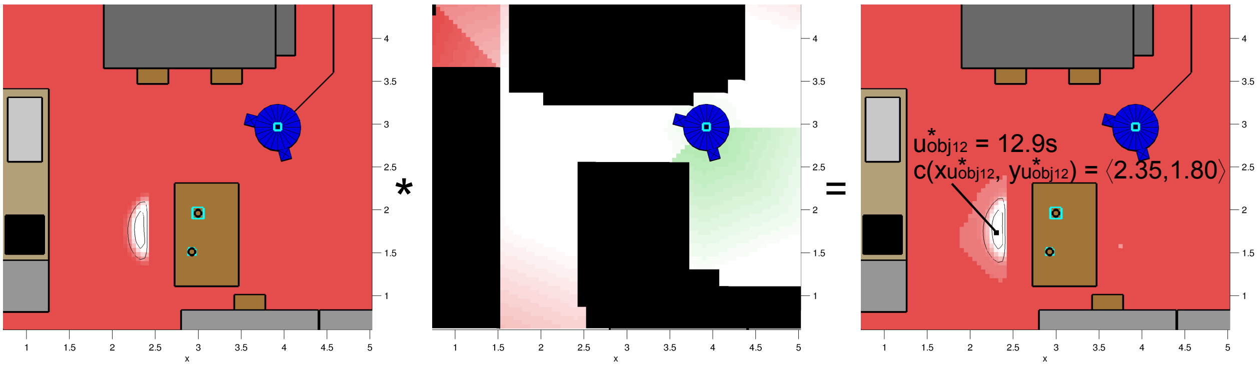


FIGURE 7.13 Left: ARPLACE probability distribution for grasping obj_1 with the right arm and obj_2 with the left arm. Center: Estimated travel time. Right: Resulting utility distribution. The grid cell with maximal utility is marked.

7.5.2 Overall Utility of Plans

The overall utility of a plan is the sum of the utilities of its plan steps. Plan A), for example, consisted of the following steps (1) Grasp obj_1; (2) Put obj_1 into dishwasher; (3) Grasp obj_2; (4) Put obj_2 into dishwasher. We can further divide the plan into the following steps.

- Move to $c(x_{u^*_{obj1}}, y_{u^*_{obj1}})$
- Grasp obj_1 with any arm
- Move to dishwasher
- Place obj_1 into dishwasher
- Move to $c(x_{u^*_{obj2}}, y_{u^*_{obj2}})$
- Grasp obj_2 with any arm
- Move to dishwasher
- Place obj_2 into dishwasher

Please note that every two consecutive steps (matching colors) involve a movement action followed by a manipulation action, and therefore can be seen as an ARPLACE query. Therefore, the overall utility of plan A) is the sum of the utilities of four ARPLACEs. This is a drawback of computing overall utility of whole plans. Increasing the lookahead of plan steps increases the number of required ARPLACEs, and leads to higher computation time. That is why computing overall utility of plans should be considered as an option that is only used when there are sufficient computational resources. The default solution to address the problem of finding manipulation places for grasping multiple objects at once is the transformational planning system. Figure 7.14 depicts the robot's trajectory in order to execute plan A) and the times that are required for each navigation action.

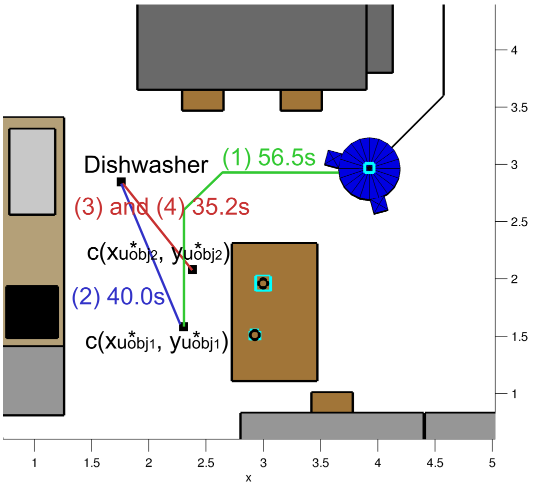


FIGURE 7.14 Navigation trajectories for executing plan A). Each path is annotated with the corresponding plan step in brackets. The following number represents the estimated navigation time for moving along the path. For clarity, corresponding paths and numbers are drawn in the same color.

According to the numbers that are presented in Figure 7.14, and the fact that one manipulation action requires an additional 20s, the overall execution time of plan A) is predicted to be

$$t(planA) = 56.5\text{s} + 40.0\text{s} + 35.2\text{s} + 35.2\text{s} + (4 \cdot 20.0\text{s}) = 246.9\text{s}$$

The grasp success probability for the two grasping tasks are 90.0% for grasping obj_1 from base position $c(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$, and 86.1% for grasping obj_2 from base position $c(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$. The overall utility of plan A) is

$$u(planA) = 0.900 \cdot 30.0\text{s} + 0.861 \cdot 30.0\text{s} + -246.9\text{s} \cdot 0.1 = 28.1\text{s}$$

Figure 7.15 depicts an overview of the relevant values for all plans including their overall utilities. There are several observations that can be made. Overall, plan E) for grasping both objects at once has the highest utility. As expected, its execution time is significantly smaller than for any other plan. However, the sum of its grasp success probabilities is the lowest with 84.7% for grasping obj_1 and 71.8% for grasping obj_2. This was also expected because plan E) has to find a manipulation place that is a good compromise for grasping both objects. Plan C) and D) have significantly lower execution times than plan A) and B) because both objects are grasped one after the other. This saves two navigation actions. One navigation action towards the dishwasher and one back to the table. Moreover, plan C) and D) have a lower sum of grasp success probabilities than plan A) and B). This is because plan A) and B) can grasp both objects from the perfect base position, which proposes to use the right arm for both objects. This is not possible in plan C) and D), because when performing the second grasping task, one gripper is already occupied. So the robot has to decide whether to perform the first or the second grasping action with the suboptimal left arm. In this scenario, the utility for grasping obj_1 with the left arm and obj_2 with the right arm is higher than vice versa.

Plan	$p(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$	$p(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$	$t(plan\circ)$	$u(plan\circ)$
A)	0.900	0.861	247.0s	28.1s
B)	0.900	0.861	246.6s	28.2s
C)	0.847	0.861	181.0s	33.2s
D)	0.847	0.861	192.6s	32.0s
E)	0.847	0.718	133.8s	33.6s

FIGURE 7.15 Grasp success probabilities, estimated execution time, and overall utility for different plans. ○ refers to the plan as specified in the left column.

Based on the overall plan utilities, the robot decides to grasp both objects at once and moves towards $c(x_{u^*_{obj_{12}}}, y_{u^*_{obj_{12}}})$.

7.6 Handling Sudden Changes

While moving, the robot encounters a chair that was previously hidden behind the kitchen table. The chair exactly blocks the robot's current navigation goal. This leads to a dramatic change in the ARPLACE utility distributions. The utility distributions before and after discovering the chair are depicted in Figure 7.16.

It can be seen that the chair significantly changes all utility distributions and additionally changes the most promising base position for all manipulation actions. The proposed base position for grasping obj_1 changes from $\langle 2.35\text{m}, 1.60\text{m}\rangle$ to $\langle 2.35\text{m}, 1.15\text{m}\rangle$, and leads to a drop of grasp success probability from 91.8% to 66.8%. The estimated travel time increases from 41.9s to 52.3s. The proposed base position for grasping obj_2 changes from $\langle 2.35\text{m}, 2.00\text{m}\rangle$ to $\langle 3.10\text{m}, 2.65\text{m}\rangle$. This is the first time that ARPLACE proposes to grasp one of the objects from another table edge than te_1. Although grasp success probability dropped from 98.2% to 81.6%, the estimated travel time got shorter and is estimated to be 13.5s now compared to 37.4s previously. In fact, the robot's current base position is very close to $c(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})\ '$. The impact of the discovered chair on the base position for grasping both objects at once is most striking, because the chair exactly blocks the area that consists the most promising base positions. Grasp success probability drops from 83.3% for grasping obj_1 with the right arm

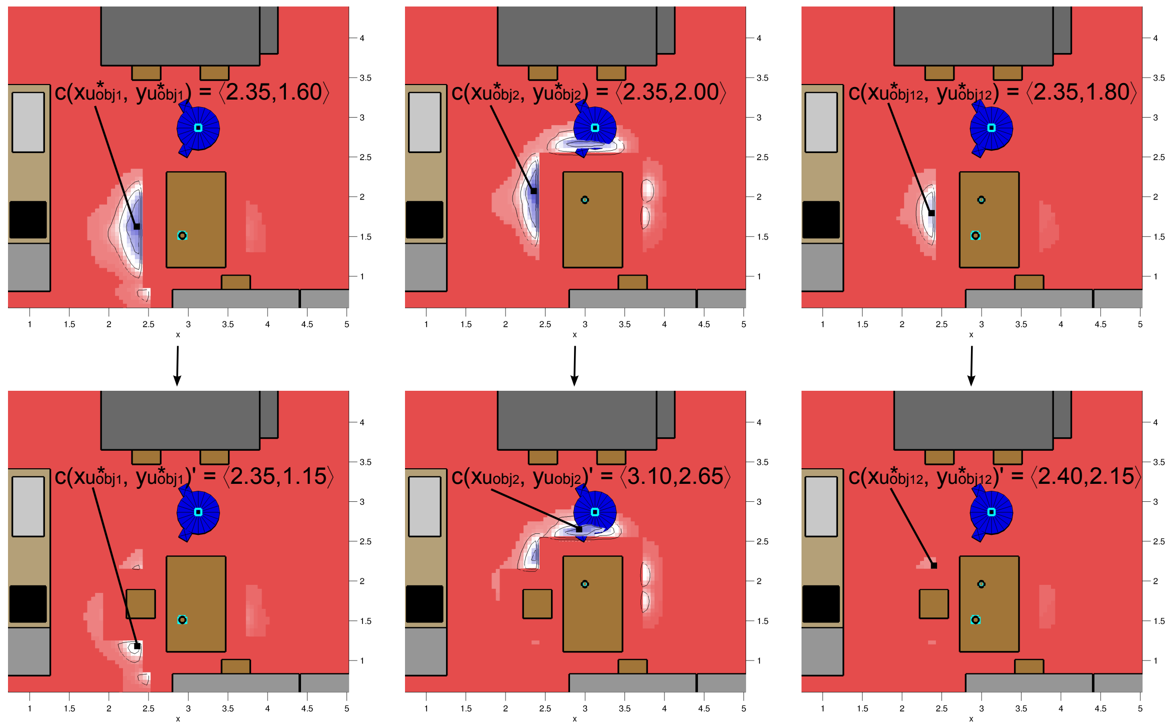


FIGURE 7.16 Utility distributions before (top image row) and after (bottom image row) the robot discovered the chair. Left: Utility distributions for grasping obj_1. Center: Utility distributions for grasping obj_2. Right: Utility distributions for grasping both objects from the same base position. The grid cells with maximum utility are marked in each plot. Grid cells that are labeled without a trailing ' refer to places before discovering the chair, and places with a trailing ' refer to places after discovering the chair.

and 83.6% for grasping obj_2 with the left arm to 13.7% and 12.5% respectively. As a result, the newly proposed base position $\langle 2.40\text{m}, 2.15\text{m}\rangle$ has a utility value of rather low -2.8s as opposed to 16.9s previously.

Figure 7.17 depicts values of the most promising base positions before and after discovering the chair. Each row presents values of a base position with maximum utility that is depicted in Figure 7.16.

After determining the new promising manipulation places, the overall utilities of plan A) - E) are computed. Figure 7.18 depicts the corresponding resulting values. It can be seen that plan D) is preferred with a utility of 21.8s. This requires the robot to move to $c(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})\,'$ in order to grasp obj_2 with the right arm, then move to $c(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})\,'$ in order to grasp obj_1 with the left arm, and finally move to the dishwasher and put down both objects. ARPLACE transfers $c(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})\,' = \langle 3.10\text{m}, 2.65\text{m}\rangle$ as the new goal position to the navigation system.

Place	$c(x_{u^*_{obj_\circ}}, y_{u^*_{obj_\circ}})$	$p(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$	$p(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$	$t(x_{u^*_{obj_\circ}}, y_{u^*_{obj_\circ}})$	$u^*_{obj_\circ}$
obj_1	$\langle 2.35\text{m}, 1.60\text{m}\rangle$	0.918 (R)	-	41.9s	23.3s
obj_2	$\langle 2.35\text{m}, 2.00\text{m}\rangle$	-	0.982 (R)	37.4s	25.7s
obj_{12}	$\langle 2.35\text{m}, 1.80\text{m}\rangle$	0.833 (R)	0.836 (L)	39.4s	16.9s
obj_1'	$\langle 2.35\text{m}, 1.15\text{m}\rangle$	0.668 (L)	-	52.3s	14.8s
obj_2'	$\langle 3.10\text{m}, 2.65\text{m}\rangle$	-	0.816 (R)	13.5s	23.1s
obj_{12}'	$\langle 2.40\text{m}, 2.15\text{m}\rangle$	0.137 (R)	0.125 (L)	35.7s	-3.1s

FIGURE 7.17 Manipulation places and relevant values before and after the robot discovers the chair. $c(x^*_{u_{obj.}}, y^*_{u_{obj.}})$ indicates the proposed base position. $p(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$ and $p(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$ indicate grasp success probability for the target objects. (R) and (L) in these columns represent whether grasp success probability refers to a grasping action with the *R*ight or *L*eft arm. $t(x^*_{u_{obj_\circ}}, y^*_{u_{obj_\circ}})$ refers to estimated travel time, and $u^*_{obj_\circ}$ is the resulting utility value. '$\circ$' refers to the object as specified in the left column.

Plan	$p(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})$	$p(x_{u^*_{obj_2}}, y_{u^*_{obj_2}})$	$t(plan\circ)$	$u(plan\circ)$
A)	0.668	0.816	292.6s	15.3s
B)	0.668	0.816	249.0s	19.6s
C)	0.668	0.816	239.0s	20.6s
D)	0.668	0.816	227.4s	21.8s
E)	0.137	0.125	150.1s	-7.1s

FIGURE 7.18 Grasp success probabilities, estimated execution time, and overall utility for different plans. '$\circ$' refers to the plan as specified in the left column.

7.7 Performing the Task

Because no further observations have an impact on the proposed base position, the robot grasps obj_2 from $\langle 3.10\text{m}, 2.65\text{m}\rangle$ with the right arm. Subsequently, the robot moves towards $c(x_{u^*_{obj_1}}, y_{u^*_{obj_1}})' = \langle 2.35\text{m}, 1.15\text{m}\rangle$ in order to grasp obj_1.

The robot moves around the chair and turns towards the table in order to grasp obj_1, which the robot believes is a glass although it is a cup. After finishing the turning motion, the vision system detects a handle at obj_1 that was previously hidden by the cup's body. The robot is now sure that obj_1 is a cup and the belief state changes from $typ_{obj_1} = \langle 0.15, 0.85\rangle$ to $typ_{obj_1} = \langle 1.0, 0.0\rangle$. Because a cup is grasped from the side at its handle, the ARPLACE utility distribution changes as well. Figure 7.19 depicts the ARPLACEs before and after observing the cup's handle.

Although the proposed base position changes only insignificantly from $\langle 2.35\text{m}, 1.15\text{m}\rangle$ to

$\langle 2.25\mathrm{m}, 1.15\mathrm{m}\rangle$, grasp success probability shrinks from 82.8% to 69.1%. The reason is that the cup's handle puts a kinematic constraint on the arm that inhibits the robot to use the left arm for grasping that was preferred previously. The robot has to use its right arm for grasping and stretch it far to the right.

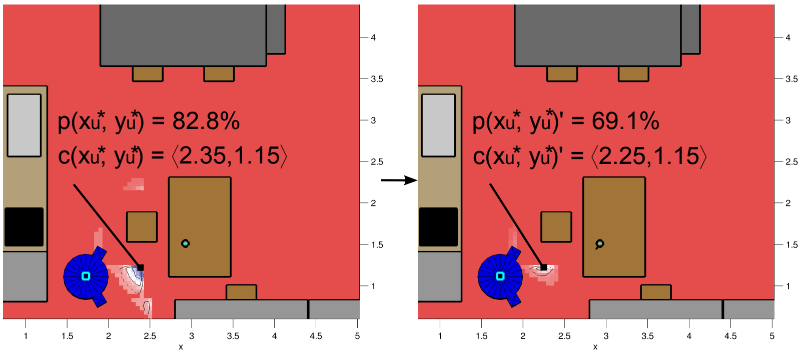


FIGURE 7.19 ARPLACE utility distributions before (left image) and after (right image) the robot discovered the cup's handle.

Because the right arm is currently occupied by obj_2, the robot puts obj_2 into the dishwasher, then moves to the proposed base position, grasps obj_1 with the right arm, and moves back to the dishwasher.

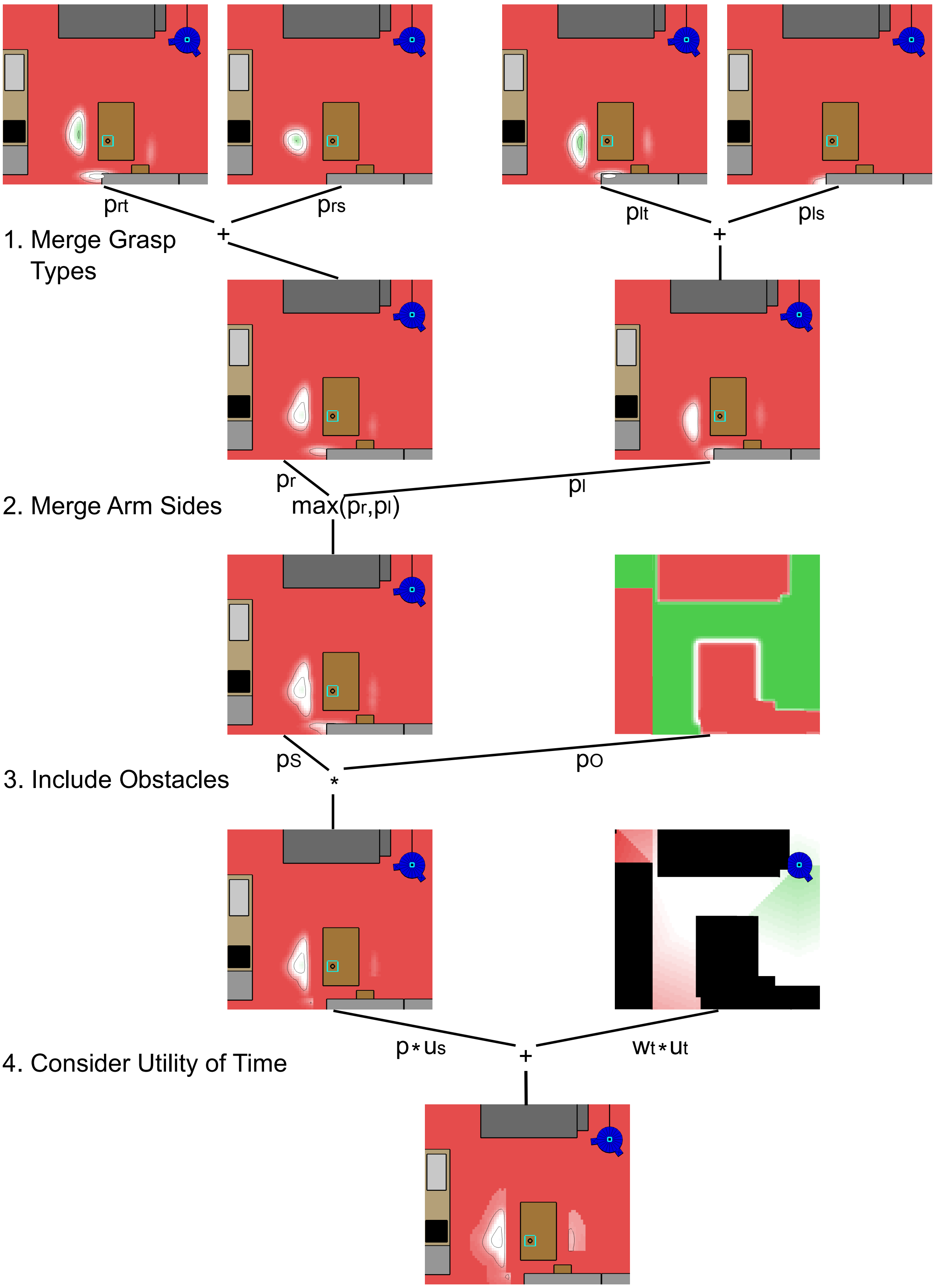


FIGURE 7.20 Steps for computing a utility-based ARPLACE.

CHAPTER 8

Conclusion and Future Research

Mobile manipulation is a challenging task and involves complex decision making. But it is highly desirable to develop robust manipulation skills, because it is one of the enabling techniques for building truely autonomous robots. The ability to find optimal manipulation places is one of the decisions that have to be made and it is an important one because it sets the context under which other manipulation subsystems such as motion planning, trajectory generation, or manipulator control have to perform the manipulation action. Finding manipulation places that are tailored towards the robot's skills makes the task of these subsystems as easy as possible and leads to more robust manipulation.

In this dissertation thesis we presented the ARPLACE framework which is a compact control program for robot positioning in mobile manipulation tasks. Section 8.1 summarizes the contributions, gives a short overview of the computational steps and presents results that were described in the thesis. We discuss directions that are promising for future research in section 8.2.

8.1 Summary

The framework of Action-Related Places introduces a new way to address the problem of finding manipulation places. It adopts paradigms from cognitive motor control and bayesian modeling in order to naturally address challenges that occur in mobile manipulation. As a result, the ARPLACE framework can adapt to different task contexts and is able to handle sensor information that is uncertain and arriving continuously.

The ARPLACE framework represents manipulation places in internal models that are called Generalized Success Model. Generalized Success Models are learned through experience-based learning, similar to how humans develop internal models. The approach of experience-based learning is able to capture robot skills that are hard to discover and represent analytically. The Generalized Success Model is used online to compute an ARPLACE probability distribution that represents grasp success probability for all base positions. According to the bayesian coding hypothesis “the human brain represents sensory information probabilistically, in the form of probability distribution”. So ARPLACE probability distributions are a natural and cognitive way of decisions making. They are also an efficient approach to find the optimal manipulation place as the one that maximizes grasp success probability. We chose to use a point distribution model as the basis for the Generalized Success Model because it is compact and can be queried fast. This allows to update ARPLACE probability distributions when new sensor information arrives and enables least commitment planning. In the following we summarize this thesis by chronologically presenting the computational steps of the ARPLACE framework.

The first step is to develop an internal model of successful manipulation places. Therefore, a target object is placed at various positions on a table. In simulation the robot tries to grasp the target object from various manipulation places, and stores whether the manipulation action succeeded or failed. Support vector machines generalize over the training data. One SVM model is learned for every distinct pose of the target object, and a classification boundary is extracted. A classification boundary represents manipulation places that lead to successful manipulation for a certain, discrete object pose. The classification boundaries are used as input for learning a point distribution model that interpolates between object poses. The resulting point distribution model is called Generalized Success Model and is a compact, yet precise model of successful manipulation places. The Generalized Success Model represents manipulation places that are not only kinematically promising, but places that have led to successful manipulation in actual manipulation tasks. The approach of experience-based learning assures that the Generalized Success Model is tailored towards the robot’s skills.

Objects are mainly grasped either by approaching them from the side or from the top. That is why we learn separate Generalized Success Models for these two types of grasps. We evaluated the resulting Generalized Success Models and found that they generalize well in that they capture 96% of the deformation energy for side grasps with two deformation modes. When considering grasps from the top, then the first deformation mode already captures 94% of the deformation energy and two deformation modes capture 99% of the deformation energy. The process of gathering training data is time consuming with approximately 50 seconds for every simulation experiment. However, it can be done offline, has to be done only once per robot, and the Gazebo simulator allows to script the training process. Moreover, using geometric knowledge and the capability map enabled us to exclude 84% of the initially planned experiments. Biasing exploration with human activity data further reduced the time for gathering training data. Learning Generalized Success Models was described in chapter 3.

Generalized Success Models are used online in order to compute ARPLACE probability distributions. An ARPLACE discretizes space into grid cells where each cell represents the predicted probability of successfully grasping the target object when the robot performs the grasping action from a base position within this grid cell. State estimation uncertainties of the robot into the target object's pose or into its own base position are explicitly taken into account by performing Monte Carlo simulation and probabilistic conditioning. A thorough evaluation was performed in the Gazebo simulator. The result was that manipulation places that were proposed by the ARPLACE framework outperformed manipulation places that were proposed by the benchmark strategy in a statistically significant way. The superiority of ARPLACE to the benchmark strategy increased when state estimation uncertainty got higher. ARPLACEs are a valuable source of information for high level planning systems because they can help to decide if additional exploration is required, if the task should be aborted because it is difficult, or what to do when a manipulation action fails unexpectedly. A performance analysis showed that the computation of a simple ARPLACE takes 40-50 milliseconds on a laptop computer. This is fast enough for performing least-commitment planning. Computing ARPLACE probability distributions was described in chapter 4.

Chapter 5 extended the concept of Action-Related Places and presented how to compute more advanced ARPLACEs. It was shown how easy it is to add additional features. Many of the presented extensions are based on the operation of computing multiple ARPLACE probability distributions and merging them. The computation of multiple ARPLACEs requires additional computation time but scales very well when it is distributed to multiple CPUs. The merging operation itself can be performed very fast because merging ARPLACEs is done by simply multiplying probability values of corresponding grid cells. It was shown how computing ARPLACEs for every table edge can lead to multi-modal ARPLACE distributions, how ARPLACEs are determined when the robot is not certain about the target object's type, or how an ARPLACE that was computed for grasping with the right arm can be transformed into an ARPLACE for grasping with the left arm. Obstacles have to be taken into account as well. In order to achieve this, a obstacle probability distribution is computed and merged with the original ARPLACE probability distribution. Another useful feature of ARPLACEs is that manipulation places for grasping multiple objects at once can be computed easily by computing separate ARPLACEs for each object and merging them. An evaluation showed that applying transformational planning for manipulating multiple objects at once reduces the average execution time from 48 seconds to 32 seconds. The chapter closed by describing how Action-Related Places are modified when a manipulation action failed.

ARPLACEs are usually based on grasp success probability. However, grasp success may not be the robot's only concern. In the presence of humans, a robot should prefer to stay within the humans' field of view. If a task is urgent, performing the task as quickly as possible has priority, and if the battery is low saving energy becomes a vital goal. A utility framework was presented in chapter 6 that generalized the framework of Action-Related Places in order to take multiple, potentially conflicting task goals into account. Utility-based ARPLACEs can be applied to a broader range of tasks and goals. The utilities we considered in this thesis were travel time and utility of successful grasping. A detailed evaluation analyzed the differences between probability-based and utility-based ARPLACEs. Manipulation places that were proposed by utility-based ARPLACEs had an average grasp success probability of 89% and an average travel time of 41.7 seconds. Manipulation places that were proposed by probability-based ARPLACEs had an average grasp success probability of 95% and an average travel time of 46.0 seconds. Overall, the utility-based ARPLACEs traded 6% of grasp success probability in order to save 9% of travel time.

A case study in chapter 7 presented a complex scenario where the robot had to perform the task "clean the kitchen table". The primary goal of the case study was to provide the reader with an intuition for how ARPLACEs behave and what manipulation places will be proposed in complex situations. First, the process of finding optimal manipulation places was described thoroughly. It was especially emphasized how new sensor data can change ARPLACE distributions. Better state estimations, for example, may lead to manipulation places with higher grasp success probability, discovering additional objects may allow to grasp multiple objects at once, and detecting obstacles can block manipulation places that seemed to be promising before. The case study also discussed how the problem of grasping multiple objects at once can be addressed within the utility-based framework.

8.2 Open Challenges and Future Research

The ARPLACE framework is applicable to a wide range of tasks and goals and has shown to propose promising base positions even in complex scenarios. Some problems, however, remain.

Until now we restrict the robot to *face the table perpendicularly* when approaching it. Although we believe that overall this is the most useful orientation, being able to compute manipulation places for any kind of robot orientation would certainly enable the robot to grasp objects that cannot be grasped otherwise. An example is depicted in Figure 8.1. This generalization of the ARPLACE framework comes with two drawbacks. First, the number of experiments for learning Generalized Success Models increases because training data has to be gathered for multiple robot orientations. The second drawback is that it is unsure whether the point distribution model can handle additional parameters. Although we have not tried this yet, we doubt that the point distribution model would generalize well and believe that new representations of the internal model and new learning techniques have to be found.

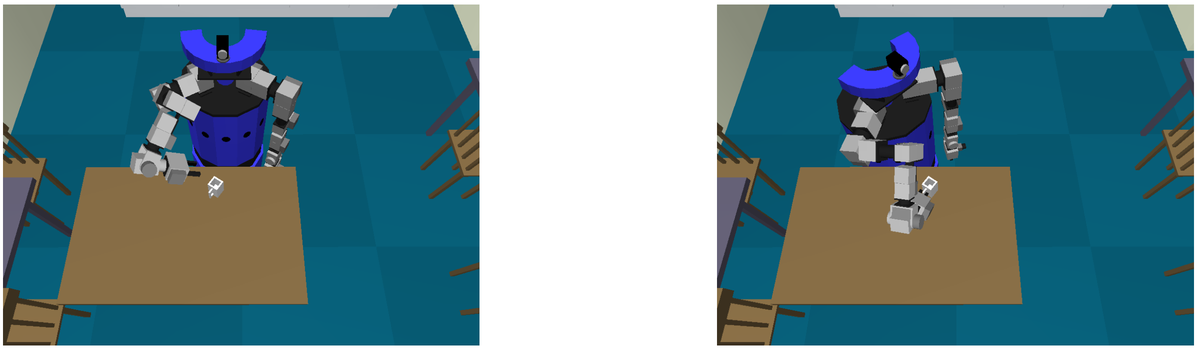


Figure 8.1 Left: Robot approaches the table perpendicularly. The target object is oriented in a way the makes it impossible for the robot to grasp it. Right: The robot changes its orientation so that grasping the object becomes possible.

Computational efficiency of utility-based ARPlaces is also an issue. Although the computation of ARPlace probability distributions is fast and can be computed within milliseconds, the computation of complex ARPlaces that include many obstacles is slower, although well below 500 milliseconds. The first possibility is to switch from the current Matlab implementation to a C++ implementation. The second possibility is to avoid to compute ARPlace distributions for all grid cells, but to tailor them towards grid cells from where the target object is theoretically within reach. The use of the capability map is currently limited to the offline part of the ARPlace framework. Using the capability map to determine the grid cells from where the target object is theoretically within reach and aborting computations after the ARPlace is computed for all such grid cells is reasonable and avoids unnecessary computation. Because ARPlace computations scale very well the third possibility is to distribute computations among multiple CPU's. All three possibilities can be used in conjunction.

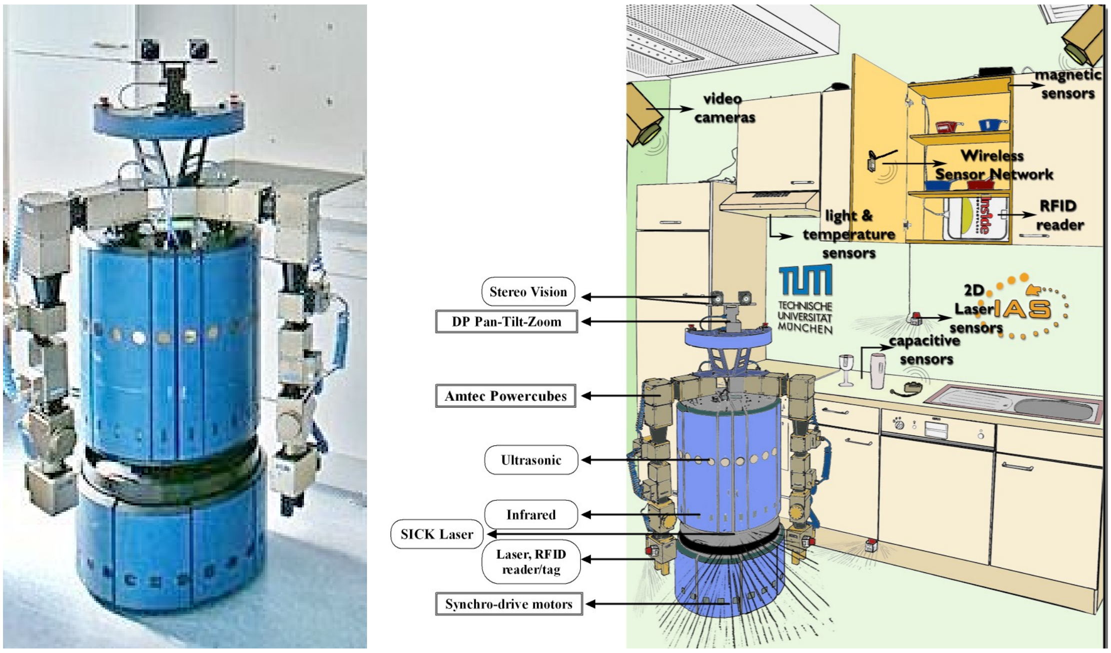
The ARPlace framework has been evaluated on the real B21r robot. Unlike in simulation it was not possible to achieve a statistically significant improvement of successful manipulation. However, this does not mean that the ARPlace framework is not working properly. The challenge is that the ARPlace system can only be evaluated in fully functional robot systems. The B21r robot is not yet robust enough to be able to exploit the benefits of the ARPlace framework. Especially the localization system's estimation about the angular orientation of the robot is critical, and Player's AMCL algorithm is not yet able to provide satisfying estimations.

It is desirable to build a mobile manipulation platform that is robust enough to enable proper evaluation of the ARPLACE framework.

Appendix A

Robot Platform

In this section, we describe the hardware and software components of our robot system. We use a B21r mobile robot from Real World Interfaces (RWI) that is visualized in Figure A.1.



(a)

(b)

FIGURE A.1 (a) Real B21r robot. (b) Schematic of sensors and actuators of the robot and its environment.

It is a cylindrical robot with a radius of 0.25m (0.35m in directions, where the arms overcome the base) and a height of 1.4m. The robot features a 4-wheel synchronous drive for locomotion that enables it to move forward and backward, and to turn around its center axis to the left and to the right. The maximum translational speed is $0.9\frac{m}{s}$ and the maximum rotational speed is $167\frac{o}{s}$. In order to avoid severe injuries the translational speed is limited by hand to $0.1\frac{m}{s}$ and the rotational speed is limited to $6\frac{o}{s}$. The arms are mounted with an offset of 12.1cm to the robot's base center as can be seen in Figure A.2.

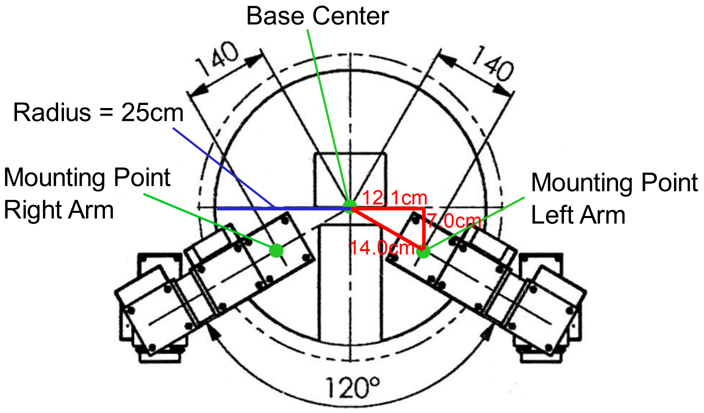


FIGURE A.2 B21r robot seen from top. The blue line visualizes the robot's radius of 25cm. The red rectangle depicts that the arms are mounted 12.1cm to the left and right of the robot's base center.

A.1 Arm Kinematics

The robot is equipped with two 6-DOF Powercube lightweight arms from Amtec Robotics that is now part of Schunk. Schematic drawings of how the links are connected are depicted in Figure A.3. All joints are rotational joints. Please note that the manipulator itself consists of the first six joints, which are labeled with the indices '0' to '5' in Figure A.3. The joint with subindex '6' corresponds to the gripper which is a single translational joint. The corresponding Denavit-Hartenberg parameters are depicted in Figure A.4.

Despite their human-like length, which is 0.84m when including the gripper, the arms are more limited than human arms mainly because the dextrous workspace is smaller. Zacharias et al. (2008) present an analysis of the workspace structure of the Powercube arms. The arms are equipped with parallel grippers that have rubber foam attached to their inside for increasing the contact area and friction between the grippers and the target object. The payload of each arm is approximately 2kg.

A.2 Perception

The vision system of the robot consists of two high dynamic range cameras (Basler Scout 1390fc, resolution: 1390x1038) forming a stereo setup. The setup is mounted on a PTU-46 pan-tilt unit from Directed Perception installed on the top of the robot.

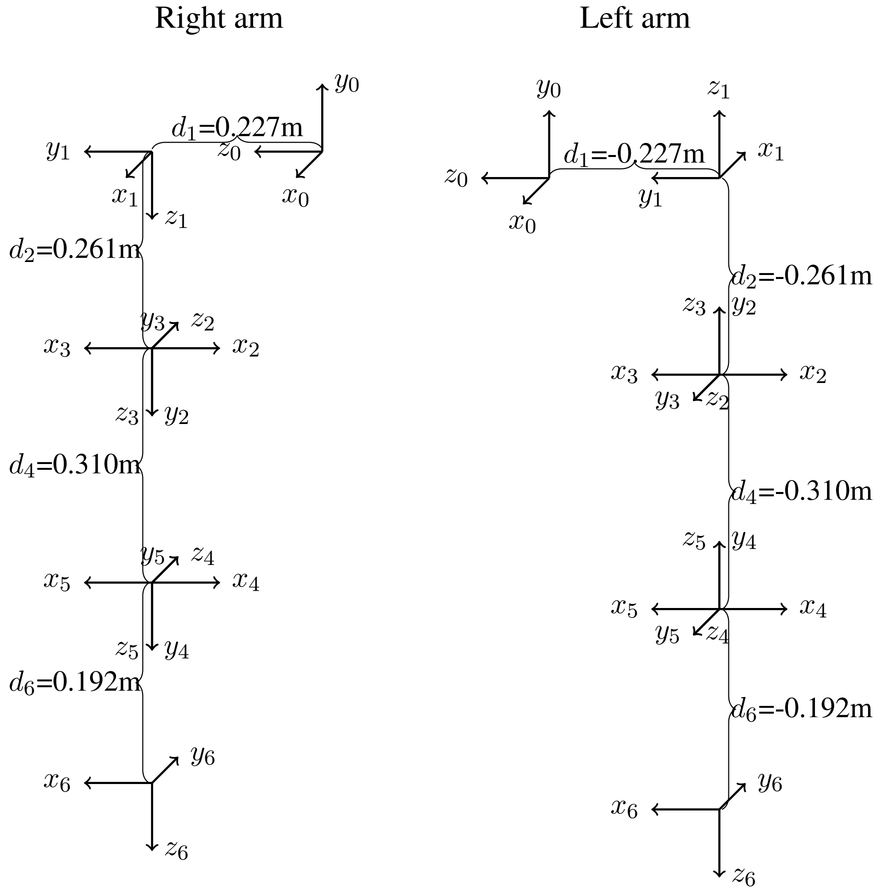


FIGURE A.3 Drawing of the right and left 6-DOF Powercube arm. The joints are numbered from '0' to '5', and the x, y, and z-axis of every joints' coordinate frame is shown. The z-axis defines the rotational axis of a joint. The values for d_i indicate the length of a link from joint i to joint $i+1$.

Localization is done using a Sick LMS 200 laser range scanner with a field of view of 180° to the front. It is mounted in the center of the robot at a height of 0.35m. The laser data is integrated with the odometry from the wheels to perform particle-based Adaptive Monte-Carlo Localization (AMCL).

In order to perform everyday manipulation tasks, the B21r robot uses global and local reference frames. The locations where the robot operates are described in global coordinates, making it necessary to localize the robot within the kitchen. Once the robot has correctly navigated to the goal base position, local perception with our vision system is used to acquire a more exact scene description. For example, if the robot has to pick up a cup from a table, it will obtain the global coordinates of the cup, navigate there, and then use its vision system in order to obtain a more precise estimation of the cup's pose relative to the robot's base.

Right arm						
	θ	d	a	α	θ_{min}	θ_{max}
${}^{0}T_1$	0°	0.227	0	90°	−270°	270°
${}^{1}T_2$	−90°	0.261	0	90°	−270°	270°
${}^{2}T_3$	180°	0.000	0	90°	−135°	135°
${}^{3}T_4$	180°	0.310	0	90°	−270°	270°
${}^{4}T_5$	180°	0.000	0	90°	−120°	120°
${}^{5}T_6$	0°	0.192	0	0°	−270°	270°

Left arm						
	θ	d	a	α	θ_{min}	θ_{max}
${}^{0}T_1$	180°	-0.227	0	90°	−270°	270°
${}^{1}T_2$	−90°	-0.261	0	90°	−270°	270°
${}^{2}T_3$	180°	0.000	0	90°	−135°	135°
${}^{3}T_4$	180°	-0.310	0	90°	−270°	270°
${}^{4}T_5$	180°	0.000	0	90°	−120°	120°
${}^{5}T_6$	0°	-0.192	0	180°	−270°	270°

FIGURE A.4 Denavit-Hartenberg parameters for defining the kinematic chain of the Powercube manipulators. The values for θ specify the angle for the home position of the arm. θ_{min} and θ_{max} specify joint limits with respect to the home position.

A.3 Computational Framework

An overview of the robot's computational framework is depicted in Figure A.5. The focus is on processes, not on models they operate on. The framework uses a variety of subsystems that communicate over two middleware frameworks: Player from the Player/Stage/Gazebo project as described by Gerkey et al. (2003), and YARP as described by Metta et al. (2006).

Player provides a variety of hardware drivers and is mainly used for low-level communication with the robot such as receiving laser sensor data, controlling the manipulators, communicating with the navigation system, and moving the pan/tilt unit of cameras. Player also comes with some standard robot algorithms. We currently use the *AMCL* algorithm for localization, *pmap* for map building, and the *Wavefront Planner* for global path planning. We added our own interfaces and drivers where necessary. For instance, we implemented an interface to the kinematics and dynamics library *Orocos-KDL* as described by Bruyninckx et al. (2003), a navigation controller, and a driver that makes the motion planning library *MSL* available for collision-free path planning.

In Figure A.5 all processes that are based on Player modules or external libraries are indicated by a white background. YARP on the other hand is used for communication between higher level systems. YARP and Player coexist peacefully, and we built middleware bridges that allow to send messages to both middleware frameworks.

For debugging and efficient data collection purposes, we also use the Gazebo simulator. Gazebo is a high-quality 3D multi-robot simulator for indoor and outdoor environments that uses the Open Dynamics Engine library for a proper simulation of rigid body physics. The simulator is especially useful for gathering large training data sets required for learning action models.

The belief state plays an important role in our robot system. In most systems that use plan-

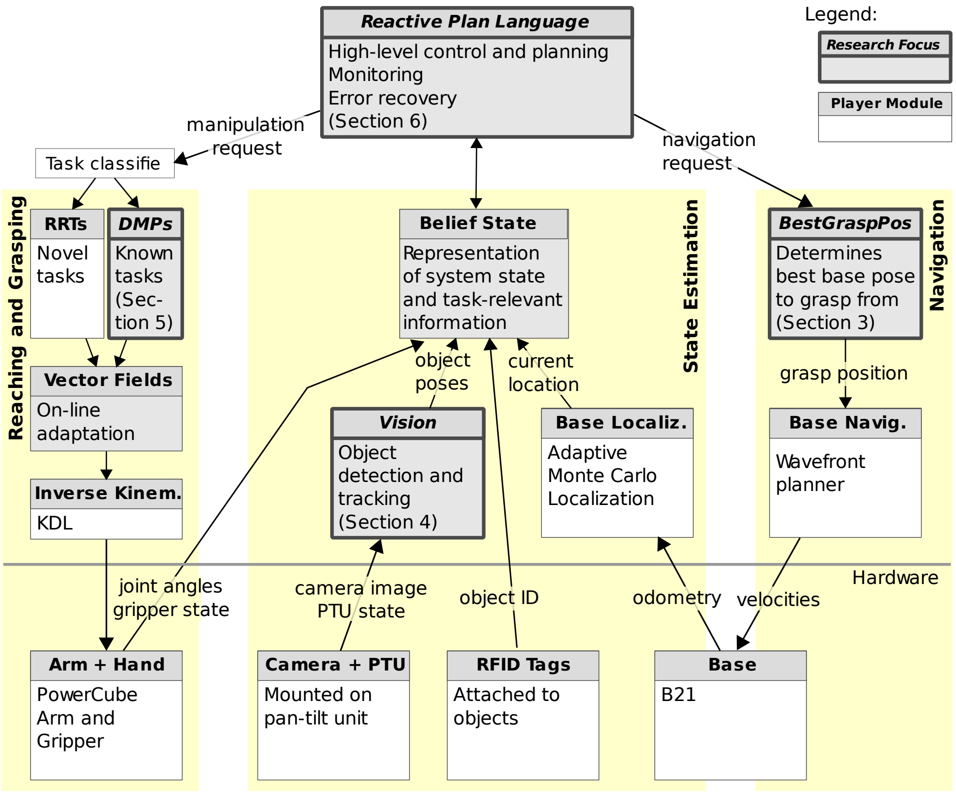


FIGURE A.5 Overview of the robot's computational framework. Low-level control loops are not depicted for clarity.

based control, the belief state is limited to abstract concepts. This simplification makes planning tractable. However, it often abstracts away from aspects that are relevant in order to understand the behavior of the robot. For a flexible adaptive planning system, it is necessary to know what caused a failure on a lower level. This information is mandatory for the robot to analyse failures and for changing future plans in order to forestall failures. Our belief state therefore receives all information from state estimation modules, and the planner chooses when and at what level of abstraction information from the belief state is required.

Appendix B

Modeling through Experience-Based Learning

In this section we discuss the advantages of finding promising manipulation places through experience-based learning instead of modelling them analytically. Section B.1 will argue that learned models are better suited to capture behavior of complex systems than analytical modelling, especially when multiple subsystems interact. We present examples where robots have learned a model with greater accuracy than could be programmed by hand. Section B.2 explains how learning enables a robot to adapt to hardware upgrades and to changing environments. Section B.3 discusses that learning is necessary when robots have to deal with uncertainty.

B.1 Building Models of Tasks

According to Silvert (2001) "one of the main roles of a professional modeler is to apply quantitative reasoning to observations about the world, in the hope of seeing aspects that may have escaped the notice of others". Traditionally, it has been the robot designers who have reasoned about *their* observations of the robot's design and behavior, and constructed models based on that. Because robots and the domains they act in are becoming more and more complex, it will become very difficult to explicitly model all interactions of the robot with the world. The alternative is to have the robot perform quantitative reasoning about, and learn from its observations of the world. This enables the robot to acquire aspects of models that the designer did not take into account.

Examples where robots have learned a model with greater accuracy than could be programmed by hand are kinematic and dynamic models (Peters, 2007), models of motor primitives (Ijspeert et al., 2002), color models (Stone et al., 2006) and model fitting applications in computer vision (Williams et al., 2005; Wimmer et al., 2008). Because all these models are fundamental to manipulation, Kemp et al. (2007) claim that "it seems almost inevitable that learning will play an important role in robot manipulation".

Another issue in modeling is the complexity of current robots. Integrated robot systems consist of dozens of sensors and actuators, and a multitude of perception and motor skills to enable the execution of complex tasks. It is perhaps unavoidable that robots are becoming as complex as the natural systems whose tasks they are trying to perform. Currently these systems are designed as a collection of subsystems, where each subsystem has a local model. According to Ghallab (2008) "a modular design of a complex artifact develops only local models that are combined on the basis of some composition principle of these models; it seldom provides global behavior models". Beetz and Belker (2000) summarize the difficulty of analytically specifying models for navigation actions: "Navigation behavior is the result of the subtle interplay of many complex factors. These factors include the robot's dynamics, sensing capabilities, surroundings, parameterizations of the control program, etc. It is impossible to provide the robot with a deep model for diagnosing navigation behavior."

Rather than *computing* global behavior by composing local models, robots could *observe* their actual global behavior, and derive models from these observations. In doing so, the need to compose local models is avoided, and the focus can be shifted on that which matters: modelling the actual global behavior in order to predict and improve future behavior.

For instance, consider the mobile manipulation example from the introduction. It is desirable to have a global model that quickly predicts from which manipulation place a grasping action will succeed. Although excellent tools for modeling the arm kinematics exist, they usually model one subsystem and do not suffice to describe the behavior of the global system. Global behavior depends on many other subsystems and factors such as arm dynamics, joint friction, controllers, motion primitive representation, navigation routines, specific skill parameterizations, high-level planner interventions, robot localization, object tracking, etc. All these issues are not important when capturing robot workspace structure as described by Zacharias et al. (2007), but do become critical when physically acting in the real world.

In general, the best strategy may be to use analytic models when they are available and use learning when these models do not suffice to model the overall system accurately enough.

B.2 Adaptation to Changes in Complex Environments

Infants are born quite helpless. Although the constant care required in early years is costly to our parents, this neotenous approach of the human species has a clear advantage: the less a human is pre-programmed for a certain environment *before* birth, the more it is possible to adapt to the environment that is encountered *after* birth. The amount of perception, action and communication skills humans acquire throughout their life is truly striking in comparison to other animals.

The same is true for robots. Only robots that learn are able to adapt to novel environments. Programmers cannot be expected to provide robots with all possible perception and action skills that they may need in every possible environment. Or as Kemp et al. (2007) state: "Learning can also help address problems of knowledge acquisition. Directly programming robots by writing code can be tedious, error prone, and inaccessible to non-experts. Through learning, robots may be able to reduce this burden and continue to adapt once they've left the factory."

Another excellent example of adaptation to the unforeseen through learning is the 'resilient machines' approach by Bongard et al. (2006), in which robots learn and update accurate models of their own structure. By doing so continually, robots are able to model severe damages to their structure and adapt their behavior accordingly. Explicitly modeling each type of wear and tear the robot might have during its life-span, and the effects this might have on its behavior is simply infeasible.

A more positive type of change robots encounter frequently is an upgrade. Controllers used for reaching, for example, are continually improved. Instead of performing an additional analysis and remodeling phase after each upgrade, we simply gather new training data and learn a new Generalized Success Model in order to adapt to the robot's new skills. The learning approach allows the robot to *track* the changes to the system with little effort on the designer's side.

In the context of manipulation, a robot cannot be expected to have a model of each object and tool it could possibly manipulate. Katz and Brock (2007) describe how interactive perception can uncover information about the environment that would otherwise be hidden. Interactive perception enables the robot to extract kinematic and dynamic properties of novel objects by physically interacting with them and analyzing the visual sensor stream. Moreover, the robot is able to determine the appropriate use of the explored tool.

Another type of learning that humans use to add to their skill repertoire is imitation. Imitation learning speeds up the acquisition of new skills tremendously, because new skills do not need to be acquired through trial-and-error from scratch (Schaal et al., 2004). Instead of enabling robots to solve all possible tasks the goal is to give the robot the general ability to *imitate* the observed solution to a task. Examples that use this approach with great success are described by Ijspeert et al. (2002) and Calinon et al. (2007).

Adapting to complex, possibly changing environments requires the robot to learn continually during its deployment. The challenge for on-line learning algorithms is to determine which features of the world are invariant, which features change over time, and which features are relevant to a task. Whereas imitation learning helps to learn satisfactory solutions quickly, online learning during a robot's operational life requires the robot to process massive amounts of data, to be selective, and to discover the structure of the world from this data.

B.3 Dealing with Uncertainty

Kemp et al. (2007) found that explicitly designing models for robots is still the dominant approach in many robotic domains. And in situations where the world's state is known, it can perform very well. One of the earliest and economically most relevant impacts of robots has been on assembly line production. Here, the environment is extremely controlled, enabling sensors to determine the world's relevant state with high accuracy. In factories, robots are like dictators, relentlessly banishing any source of uncertainty or hidden state from their surroundings. Unfortunately, outside the factory floor robots have to deal with uncertainty in sensor signals, in the environment, in their motor commands, and in the predicted behavior of others. Moreover, partial observability complicates successful acting.

Learning enables robots to leave the factory floor. For instance by acquiring statistics from natural environments. These statistics are used to derive unobservable properties, or explicitly model sources of uncertainty and choosing the appropriate behavior based on this information. Partially Observable Markov Decision Processes (POMDPs) take this approach, and have been used successfully by Simmons and Koenig (1995) for robust navigation on real robots. Koenig and Simmons (1996b) improved the system by adjusting the probabilities of the initial Markov model by passively observing the robot's interactions with its environment. Schmidt-Rohr et al. (2008) present a current approach that deploys POMDPs in service robot applications including multi-modal human-robot interaction.

Two recent examples of learning probabilistic representations of the success of reaching and grasping are given by Detry et al. (2009) and Montesano and Lopes (2009). In their papers, the authors deem explicit models as being too imprecise to model grasp affordance, and thus use a trial-and-error learning approach to acquire this knowledge. Montesano and Lopes (2009) argue that "a key point of this type of knowledge is that it is based on the robot experience. This is important since it guarantees that the robot morphology is implicitly embedded in the learning process and, therefore, the resulting models depend on it. Examples of this dependency are abundant in the affordance literature in ecological psychology."

Bibliography

Rachid Alami, Jean Paul Laumond, and Thierry Siméon. Two manipulation planning algorithms. In *Proceedings of the workshop on Algorithmic foundations of robotics table of contents*, pages 109–125, San Francisco, USA, 1995. A. K. Peters. ISBN 1-56881-045-8.

Min Zhao Ansari and N. Hou. Mobile manipulator path planning by a genetic algorithm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 681–688, July 1992. ISBN 0-7803-0737-2.

Hajime Asama, Masatoshi Sato, Luca Bogoni, Hayato Kaetsu, Akihiro Matsumoto, and Isao Endo. Development of an omnidirectional mobile robot with 3 dof decoupling drive mechanism. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1925–1930, 1995.

Jan Bandouch and Michael Beetz. Tracking humans interacting with the environment using efficient hierarchical sampling and layered observation models. In *IEEE Int. Workshop on Human-Computer Interaction (HCI). In conjunction with ICCV2009*, 2009.

Barraquand and Jean-Claude Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 1712–1717, Cincinatti, USA, 1990. ISBN 0-8186-9061-5.

Bernard Bayle, Jean-Yves Fourquet, and Marc Renaud. A coordination strategy for mobile manipulation. In *Intelligent Autonomous Systems (IAS-6)*, pages 981–988. IOS Press, 2000.

M. Beetz and T. Belker. XFRMLearn - a system for learning structured reactive navigation plans. In *Proceedings of the 8th International Symposium on Intelligent Robotic Systems*, Reading, UK, 2000.

M. Beetz and D. McDermott. Declarative goals in reactive plans. In J. Hendler, editor, *First International Conference on AI Planning Systems*, pages 3–12, Morgan Kaufmann, 1992.

Michael Beetz. Structured Reactive Controllers. *Journal of Autonomous Agents and Multi-Agent Systems. Special Issue: Best Papers of the International Conference on Autonomous Agents '99*, 4:25–55, March/June 2001.

Michael Beetz and Henrik Grosskreutz. Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior. In *Proceedings of the Sixth International Conference on AI Planning Systems*. AAAI Press, 2000.

Michael Beetz, Freek Stulp, Piotr Esden-Tempski, Andreas Fedrizzi, Ulrich Klank, Ingo Kresse, Alexis Maldonado, and Federico Ruiz. Generality and legibility in mobile manipulation. *Autonomous Robots Journal (Special Issue on Mobile Manipulation)*, 28(1): 21–44, 2010.

Dmitry Berenson, Howie Choset, and James Kuffner. An optimization approach to planning for mobile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2008*, pages 1187–1192, May 2008.

Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. In *IEEE-RAS International Conference on Humanoid Robots*, 2007.

Dmitry Berenson and Siddhartha Srinivasa. Grasp synthesis in cluttered environments for dexterous hands. In *Robotics Science and Systems (RSS) Workshop on Robot Manipulation: Intelligence in Human Environments*, June 2008.

Dmitry Berenson and Siddhartha Srinivasa. Probabilistically complete planning with end-effector pose constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

Dmitry Berenson, Siddhartha Srinivasa, David Ferguson, Alvaro Collet Romea, and James Kuffner. Manipulation planning with workspace goal regions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009a.

Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Addressing pose uncertainty in manipulation planning using task space regions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, October 2009b.

D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1874–1879, 2006.

Francois Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1):231–240, 2004.

Robert Bohlin and Lydia E. Kavraki. Path planning using lazy prm. In *IEEE International Conference Robototics and Automation*, pages 521–528, 2000.

J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314:1118–1121, 2006.

J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

O. Brock and R. Grupen. NSF/NASA workshop on autonomous mobile manipulation (amm). Technical report, University of Massachusetts Amherst, August 2005.

Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research*, 21(12):1031–1052, 2002.

Rodney A. Brooks and Tomas Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *IEEE Systems, Man and Cybernetics*, pages 224–233, april 1985.

Herman Bruyninckx, Peter Soetens, and Bob Koninckx. The real-time motion control core of the Orocos project. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2766–2771, 2003.

Sebastian Buck and Martin Riedmiller. Learning situation dependent success rates of actions in a RoboCup scenario. In *Pacific Rim International Conference on Artificial Intelligence*, page 809, 2000.

S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298, 2007.

Stephane Cambon, Fabien Gravot, and Rachid Alami. asymov: Towards more realistic robot plans. In *International Conference on Automated Planning and Scheduling, (ICAPS 2004)*, 2004a.

Stephane Cambon, Fabien Gravot, and Rachid Alami. A robot task planner that merges symbolic and geometric reasoning. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 895–899, 2004b.

Stéphane Cambon, Fabien Gravot, and Rachid Alami. Overview of asymov: Integrating motion, manipulation and task planning. In *ICAPS Doctoral Consortium*, 2003.

Jonathan M. Cameron, Ronald C. Arkin, Douglas C. MacKenzie, Wayne J. Book, and Keith R. Ward. Reactive control for mobile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 228–235, 1993.

John Canny. Constructing roadmaps of semi-algebraic sets. *Artificial Intelligence Journal*, 37:203–222, 1988a.

John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988b.

John F. Canny and John H. Reif. New lower bound techniques for robot motion planning problems. In *28th Annual IEEE Symposium on Foundations of Computer Science*, pages 49–60, Los Angeles, USA, 1987.

H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, , and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005. ISBN 978-0262033275.

Brad J. Clement, Edmund H. Durfee, and Anthony C. Barrett. Abstract reasoning for planning and coordination. *Journal of Artificial Intelligence Research*, 28:453–515, 2007.

Tim F. Cootes, A. Hill, and Chris J. Taylor. Medical image interpretation using active shape models: Recent advances. In *14th International Conference on Information Processing in Medical Imaging*, pages 371–372, 1995a.

Tim F. Cootes and Chris J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, Wolfson Image Analysis, Imaging Science and Biomedical Engineering, Manchester M13 9PT, United Kingdom, 2004.

Tim F. Cootes, Chris J. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995b.

John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1986. ISBN 978-0201103267.

J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Transactions of the ASME. Journal of Applied Mechanic*, 22:215–221, 1955.

R. Detry, E. Baseski, M. Popovic, Y. Touati, N Krueger, O. Kroemer, J. Peters, and J Piater. Learning object-specific grasp affordance densities. In *Proceedings of the International Conference on Development and Learning (ICDL)*, 2009.

Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.

Rosen Diankov, Nathan Ratliff, Dave Ferguson, Siddhartha Srinivasa, and James Kuffner. Bispace planning: Concurrent multi-space exploration. In *Proc. Int. Conf. on Robotics: Science and Systems*, 2008.

M. Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, July 2000.

R.S. Fearing and J. M. Hollerbach. Basic solid mechanics for tactile sensing. *International Journal of Robotics Research*, 4(3):40–54, 1985.

Roy Featherstone and David E. Orin. Robot dynamics: Equations and algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 826–834, San Francisco, CA, 2000. IEEE.

Andreas Fedrizzi, Lorenz Moesenlechner, Freek Stulp, and Michael Beetz. Transformational planning for mobile manipulation based on action-related places. In *Proceedings of the International Conference on Advanced Robotics (ICAR).*, 2009.

J. Randall Flanagan, Philipp Vetter, Roland S. Johansson, and Daniel M. Wolpert. Prediction precedes control in motor learning. *Current Biology*, 13:146–150, January 2003.

Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 343–349, 1999.

M. Fox and D. Long. PDDL2.1: An extension of PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

Marc Friedman and Daniel S. Weld. Least-commitment action selection. In *Proceedings 3rd International Conference on A.I. Planning Systems*. AAAI Press, 1996.

Cipriano Galindo, Juan-Antonio Fernández-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robot. Auton. Syst.*, 56(11):955–966, 2008. ISSN 0921-8890.

Christoph Geib, K. Mourao, R. Petrick, M. Pugeault, M. Steedman, N. Krüger, and Florentin Wörgötter. Object action complexes as an interface for planning and robot control. In *Proceedings of the 2006 IEEE RAS International Conference on Humanoid Robots, Genova*, 2006.

Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage Project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*, pages 317–323, 2003.

Brian P. Gerkey and Maja J. Matarić. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3862–3868, Taipei, Taiwan, 2003.

Malik Ghallab. Modeling through machine learning in robotics. In *9th African Conference on Research in Computer Science (CARI)*, pages 25–26, Rabat,Morocco, October 2008.

James J. Gibson. *The Theory of Affordances*. John Wiley & Sons, 1977. ISBN 0-470-99014-7.

Michael Gienger, Klaus Löffler, and Friedrich Pfeiffer. Towards the the design of a biped jogging robot. In *Proceedings of the IEEE International Conference on Robotics and Automat ion (ICRA)*, pages 4140–4145, 2001.

Michael Gienger, Marc Toussaint, Nikolay Jetchev, Achim Bendig, and Christian Goerick. Optimization of fluent approach and grasp motions. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 111–117, 2008.

C. Goldfeder, M. Ciocarlie, H. Dang, and P.K. Allen. The Columbia Grasp Database. In *International Conference on Robotics and Automation (ICRA)*, 2009.

Fabien Gravot, Stephane Cambon, and Rachid Alami. asymov: A planner that deals with intricate symbolic and geometric problems. In *Proceedings of the 11th International Symposium on Robotics Research (ISRR)*, pages 100–110, 2003.

Thomas L. Griffiths, Charles Kemp, and Josh B. Tenenbaum. *The Cambridge handbook of computational cognitive modeling*, chapter Bayesian models of cognition. Cambridge University Press, 2008.

Rakesh Gupta and Mykel J. Kochenderfer. Common sense data acquisition for indoor mobile robots. In *Nineteenth National Conference on Artificial Intelligence (AAAI-04*, pages 605–610, 2004.

Peter Haddawy and Steve Hanks. Utility models for goal-directed decision-theoretic planners. *Computational Intelligence*, 14, 1993.

S. Hart, S. Ou, J. Sweeney, and R. Grupen. A framework for learning declarative structure. In *RSS-06 Workshop: Manipulation for Human Environments*, 2006.

Gerhard Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, and M. Schedl. Dlr's torque-controlled light weight robot iii - are we reaching the technological limits now? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1710–1716, 2002.

J.M. Hollerbach, M.T. Mason, and H. Christensen. A roadmap for us robotics - From internet to robotics. Technical report, Computing Community Consortium (CCC), 2009.

Randy C. Hoover, Anthony A. Maciejewski, and Rodney G. Roberts. Pose detection of 3-d objects using s2-correlated images and discrete spherical harmonic transforms. In *IEEE International Conference on Robotics and Automation (ICRA), 2008*, pages 993–998, Pasadena,USA, 2008.

John E. Hopcroft, Deborah Joseph, and Sue Whitesides. Movement problems for 2-dimensional linkages. *SIAM Journal of Computing*, 13(3):610–629, 1984.

Ronald A. Howard and J.E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume 2, pages 712–762. Strategic Decisions Group, Menlo Park, California, 1984.

David Hsu, Jean claude Latombe, and Stephen Sorkin. Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings IEEE International Symposium on Assembly and Task Planning (ISATP'99*, pages 280–285, 1999.

A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *International Conference on Robotics and Automation (ICRA2002)*, 2002.

Michael I. Jordan and Daniel M. Wolpert. *Computational Motor Control*, chapter 10, pages 371–422. MIT Press, Cambridge, 1999.

Dov Katz and Oliver Brock. Extracting planar kinematic models using interactive perception. In *RSS-07 Workshop: Robot Manipulation: Sensing and Adapting to the Real World*, Atlanta Georgia, 2007.

Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena,USA, may 2008.

Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments. *IEEE Robotics and Automation Magazine*, 14(1):20–29, 2007.

Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robotsi. *International Journal of Robotic Research*, 5(1):60ff, 1986.

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. Real-time cad model matching for mobile manipulation and grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 7-10 2009a.

Ulrich Klank, Muhammad Zeeshan Zia, and Michael Beetz. 3D Model Selection from an Internet Database for Robotic Vision. In *International Conference on Robotics and Automation (ICRA)*, 2009b.

David Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12), 2004.

Sven Koenig and Reid Simmons. Modeling risk and soft deadlines for robot navigation. In *Proceedings of the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems (AAAI Technical Report SS-96-04)*, Stanford, CA, USA, 1996a.

Sven Koenig and Reid G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2301–2308, 1996b.

Benjamin Kuipers, Patrick Beeson, Joseph Modayil, and Jefferson Provost. Bootstrap learning of foundational representations. *Connection Science*, 18:145–158, 2006.

S. Larkin. *Time and Energy in Decision Making*. PhD thesis, Oxford University, 1981.

Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

Jean-Paul Laumond, editor. *Robot Motion Planning and Control*. Lectures Notes in Control and Information Sciences 229. Springer Verlag, 1998. ISBN 3-540-76219-1.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

Steven LaValle and James Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2001.

Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, 1998.

John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1442–1447, 1991.

Stephen R. Lindemann and Steven M. LaValle. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

Stephen. R. Lindemann and Steven M. LaValle. A multiresolution approach for motion planning under differential constraints. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 139–144, 2006.

Tomas Lozano-Perez. Spatial planning: A configuration space approach, 1980.

Alexis Maldonado, Ulrich Klank, and Michael Beetz. Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010. Accepted for publication.

Dinesh Manocha and John F. Canny. Real time inverse kinematics for general 6r manipulators. In *Proceedings of the IEEE International Conference Robotics and Automation (ICRA)*, pages 383–389, 1992.

Hirose Masato and Takenaka Tooru. Development of humanoid robot ASIMO. Technical Report 1, Honda Research and Development, 2001.

Matthew T. Mason and J. Kenneth Salisbury. *Robot Hands and the Mechanics of Manipulation*. MIT Press, Cambridge, MA, 1985.

D. McDermott. A Reactive Plan Language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

D. McDermott. An algorithm for probabilistic, totally-ordered temporal projection. In O. Stock, editor, *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, Dordrecht, 1997.

David McFarland and Emmet Spier. Basic cycles, utility and opportunism in self-sufficient robots. *Robotics and Autonomous Systems*, 20, 1997.

Tad McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9 (2):62–82, 1990.

G. Metta, P. Fitzpatrick, and L. Natale. YARP: Yet Another Robot Platform. *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, 3(1), 2006.

A. Miller and P. Allen. Graspit!: A versatile simulator for grasp analysis. In *Proceedings ASME International Mechanical Engineering Congress and Exposition*, 2000.

Andrew T. Miller and Peter K. Allen. Examples of 3d grasp quality computations. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 1240–1246, 1999.

Luis Montesano and Manuel Lopes. Learning grasping affordances from local visual descriptors. In *Proceedings of the International Conference on Development and Learning (ICDL)*, 2009.

Hans Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121, March 1985.

Lorenz Mösenlechner and Michael Beetz. Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior. In *19th International Conference on Automated Planning and Scheduling (ICAPS'09).*, 2009.

P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th International Symposium on Robotics Research*, Tokyo, Japan, 1989.

Armin Müller. *Transformational Planning for Autonomous Household Robots using Libraries of Robust and Flexible Plans*. PhD thesis, Technische Universität München, 2008.

Nils J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Washington D.C., USA, May 1969.

Simon Y. Nof. *Handbook of Industrial Robotics*. Wiley, New York, USA, 2nd edition, 1999.

Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Journal of Robotics and Autonomous Systems (JRAS), Special Issue on Semantic Knowledge in Robotics*, 56(11):915–926, 2008. ISSN 0921-8890.

Kei Okada, Mitsuharu Kojima, Yuichi Sagawa, Toshiyuki Ichino, Kenji Sato, and Masayuki Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 7–12, 2006.

David E. Orin and William W. Schrader. Efficient computation of the jacobian for robot manipulators. *International Journal of Robotic Research*, 3(4):66–75, 1984.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of the International Conference on Robotics and Automation (icra2009)*, 2009.

Richard P. Paul and Hong Zhang. Robot motion trajectory specification and generation. In *2nd International Symposium on Robotics Research*, Kyoto, Japan, 1984.

Jan Peters. *Machine Learning of Motor Skills for Robotics*. PhD thesis, Department of Computer Science, University of Southern California, 2007.

Donald L. Pieper. *The Kinematics of Manipulators Under Computer Control*. PhD thesis, Stanford University, 1968.

François G. Pin and Jean-Christophe Culioli. Optimal positioning of combined mobile platform-manipulator systems for material handling tasks. *Journal of Intelligent and Robotic Systems*, 6(2):165–182, December 2005.

John H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual IEEE Symposium on Foundations of Computer Science*, pages 421–427, San Juan, Puerto Rico, 1979.

John H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *26th Annual IEEE Symposium on Foundations of Computer Science*, pages 144–154, Portland, USA, 1985.

Pierre Roduit, Alcherio Martinoli, and Jacques Jacot. A quantitative method for comparing trajectories of mobile robots using point distribution models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2441–2448, 2007.

David A. Rosenbaum, Rajal G. Cohen, Ruud G. J. Meulenbroek, and Jonathan Vaughan. Plans for grasping objects. In Mark L. Latash and Francis Lestienne, editors, *Motor Control and Learning*, pages 9–25. Springer US, 2006.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009.

A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157, 2008.

Stefan Schaal, Auke-Jan Ijspeert, and Aude Billard. *The neuroscience of social interaction*, chapter Computational approaches to motor learning by imitation, pages 199–218. Oxford University Press, 2004.

S.R. Schmidt-Rohr, M. Losch, and R. Dillmann. Human and robot behavior modeling for probabilistic cognition of an autonomous service robot. In *The 17th IEEE International Symposium on Robot and Human Interactive Communication, 2008 (RO-MAN 2008)*, pages 635–640, Munich, Germany, 2008.

Matthew D. Schmill, Tim Oates, and Paul R. Cohen. Learning planning operators in real-world, partially observable environments. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (ICAPS)*, pages 246–253, 2000.

Jacob T. Schwartz, Micha Sharir, and John E. Hopcroft. *Planning, geometry, and complexity of robot motion*. Ablex, 1987. ISBN 0893913618.

Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1999.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, December 2001. ISBN 0262194759.

Lorenzo Sciavicco and Bruno Siciliano. *Modeling and Control of Robot Manipulators*. Springer, 2000. ISBN 978-1852332211.

Homayoun Seraji. An on-line approach to coordinated mobility and manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 28–35, 1993.

Homayoun Seraji. Reachability analysis for base placement in mobile manipulators. *Journal of Robotic Systems*, 12(1):29–43, 1995.

Reza Shadmehr and Ferdinando Mussa-Ivaldi. Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, 14(5):3208–3224, 1994.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004. ISBN 0521813972.

Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008.

William Silvert. Modeling as a discipline. *International Journal General Systems*, 30(3), 2001.

Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

Jivko Sinapov and Alex Stoytchev. The boosting effect of exploratory behaviors. In *In Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, Atlanta, USA, July 2010.

Emrah Akin Sisbot. *Towards Human-Aware Robot Motion*. PhD thesis, Université Paul Sabatier, Toulouse, 2008.

Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23:874–883, 2007.

J. Q. Smith. *Decision Analysis*. Kluwer Academic Publishers, 1988. ISBN 0412275201.

Russell Smith. Open dynamics engine, 2004.

S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Schoelkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

Michael Stilman and James Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robotics (Humanoids)*, volume 1, pages 322–341, December 2004.

Michael Stilman, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Planning and executing navigation among movable obstacles. In *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS)*, pages 820–826, October 2006.

Peter Stone, Mohan Sridharan, Daniel Stronger, Gregory Kuhlmann, Nate Kohl, Peggy Fidelman, and Nicholas K. Jong. From pixels to multi-robot decision-making: A study in uncertainty. *Robotics and Autonomous Systems*, 54(11):933–43, November 2006. Special issue on Planning Under Uncertainty in Robotics.

Alex Stoytchev. Some basic principles of developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 1(2):122–130, 2009.

Freek Stulp and Michael Beetz. Refining the execution of abstract actions with learned action models. *Journal of Artificial Intelligence Research (JAIR)*, 32, June 2008.

Freek Stulp, Andreas Fedrizzi, and Michael Beetz. Action-related place-based mobile manipulation. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2009a.

Freek Stulp, Andreas Fedrizzi, and Michael Beetz. Learning and performing place-based mobile manipulation. In *Proceedings of the 8th International Conference on Development and Learning (ICDL).*, 2009b.

Freek Stulp, Andreas Fedrizzi, Franziska Zacharias, Moritz Tenorth, Jan Bandouch, and Michael Beetz. Combining analysis, imitation, and experience-based learning to acquire a concept of reachability. In *9th IEEE-RAS International Conference on Humanoid Robots*, 2009c.

Gerald Jay Sussman. *A computational model of skill acquisition*. PhD thesis, Massachusetts Institute of Technology, 1973.

Y. Takahashi, T. Takagaki, J. Kishi, and Y. Ishii. Back and forward moving scheme of front wheel raising for inverse pendulum control wheel chair robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3189–3194, 2001.

Jindong Tan, Ning Xi, and Yuechao Wang. Integrated task planning and control for mobile manipulators. *The International Journal of Robotics Research*, 22(5):337–354, 2003.

Moritz Tenorth, Jan Bandouch, and Michael Beetz. The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE Int. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS). In conjunction with ICCV2009*, 2009.

Moritz Tenorth and Michael Beetz. KnowRob — Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2009.

Lung-Wen Tsai. *Robot Analysis, The Mechanics of Serial and Parallel Manipulators*. Wiley, New York, USA, 1999.

Nikolaus Vahrenkamp, Dmitry Berenson, Tamim Asfour, James Kuffner, and Rudiger Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2009.

Nikolaus Vahrenkamp, Steven Wieland, Pedram Azad, David Gonzalez, Tamim Asfour, and Rüdiger Dillmann. Visual servoing for humanoid grasping and manipulation tasks. In *Proceedings of the 8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 406–412, Dec. 2008.

J Michael Vandeweghe, David Ferguson, and Siddhartha Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *IEEE-RAS International Conference on Humanoid Robots*, November 2007.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer MIT Press, 1995. ISBN 0387987800.

R. Vijakumar, K.J. Waldron, and M.J. Tsai. Geometric optimization of manipulator structures for working volume and dexterity. *International Journal of Robotic Research*, 5(2):91–103, 1986.

John von Neumann and Oskar Morgenstern. *The Theory of Games and Economic Behavior*. Princeton University Press, 1944. ISBN 9780691130613.

Miomir Vukobratovic and Branislav Borovac. Zero-moment-point - thirty five years of its life. *International Journal of Humanoid Robotics*, 1(1):157–173, 2004.

Masayoshi Wada and Shunji Mori. Holonomic and omnidirectional vehicle with conventional tires. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, pages 3671–3676, Minneapolis, USA, 1996.

K.J. Waldron and A. Kumar. The dextrous workspace. In *ASME Mech. Conf.*, Los Angeles, USA, 1980.

Michael P. Wellman, Matthew Ford, and Kenneth Larson. Path planning under time-dependent uncertainty. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 532–539. Morgan Kaufmann, 1995.

Oliver Williams, Andrew Blake, and Roberto Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1292–1304, 2005. ISSN 0162-8828.

Matthias Wimmer, Freek Stulp, Sylvia Pietzsch, and Bernd Radig. Learning local objective functions for robust face model fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1357–1370, 2008. ISSN 0162-8828.

Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.

Florentin Wörgötter, Alejandro Agostini, Norbert Krüger, N. Shylo, and Bernd Porr. Cognitive agents - a procedural perspective relying on the predictability of object-action-complexes (oacs). *Robotics and Autonomous Systems*, 57(4):420–432, 2009.

Y. Yamamoto and X. Yun. Unified analysis on mobility and manipulability of mobile manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1200–1206, Detroit,USA, May 1999.

Anna Yershova, Léonard Jaillet, Thierry Siméon, and Steven M. LaValle. Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3867–3872, 2005.

Anna Yershova and Steven M. LaValle. Improving motion planning algorithms by efficient nearest-neighbor searching. *IEEE Transactions on Robotics*, 23(1):151–157, February 2007.

Håkan L. S. Younes. Extending pddl to model stochastic decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling Workshop on PDDL*, pages 95–103, 2003.

F. Zacharias, Ch. Borst, and G. Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3229–3236, 2007.

F. Zacharias, Ch. Borst, and G. Hirzinger. Positioning mobile manipulators to perform constrained linear trajectories. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2578–2584, 2008.

Franziska Zacharias, Wolfgang Sepp, Christoph Borst, and Gerd Hirzinger. Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In *International Conference on Humanoid Robots (Humanoids)*, 2009.

D. J. Zhu and Jean-Claude Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7(1):9–20, 1991.

Robert Zlot, Anthony Stentz, M. Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.