Probabilistic Online Prediction of Robot Action Results based on Physics Simulation

Dissertation zur Erlangung
des Doktorgrades Dr. rer. nat.
an der Fakultät für Mathematik,
Informatik und Naturwissenschaften
Fachbereich Informatik
der Universität Hamburg

vorgelegt von **Sebastian Rockel**
Hamburg, 2015

Disputation: 30. November 2015

Gutachter:
Prof. Dr. Jianwei Zhang
Prof. Bernd Neumann, Ph.D.

The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" (I found it!) but "That's funny..."

Isaac Asimov ($\star$1920 - †1992)

Imagination ... is more important than knowledge. Knowledge is limited. Imagination encircles the world.

Albert Einstein (October 26, 1929)

Abstract

The motivation for this work is the today's abstraction gap between the high-level robot control involved in symbolic planning and the low-level, fine-grained control of mobile robot motors. To deal with the complex, changeable environments in which humans live, state-of-the-art robots need to be able to exploit common-sense knowledge (i.e. physical laws) to calculate velocity, acceleration, friction and so on. Physical prediction is typically the domain of a physics simulation, such as Gazebo.

The literature contains examples in which symbolic planning and reasoning methods are extended, such as by adding geometric or temporal extensions or by offline recording of simulated actions.

The goal of this thesis is to question how a task-planning based robot system can be improved by prediction derived from physical simulation and to provide some answers. The goal-directed approach is to integrate a realistic prediction of robot activities so as to allow these activities to be adapted before execution fails. The hypothesis, therefore, is that a system that integrates such prediction is more efficient than a comparable system that does not.

A probabilistic prediction method named "functional imagination" and a system architecture have been developed. The prediction method was integrated into a blackboard-based robot control system and a PR2 robot and Gazebo were used for its evaluation. Experiments verified that simulation is indeed accurate and close to reality. The results of three experimental scenarios in a restaurant domain supported the hypothesis that the robustness and efficiency of such a planning and execution-based system are improved by the addition of physics-based prediction.

The work concludes by summarizing its findings, limitations and possible further research directions and with a future perspective on how parallel and cloud computing will affect the field of robotics in the context of simulation.

Zusammenfassung

Der Ansatz dieser Arbeit ist motiviert von der „Abstraktionslücke“ zwischen heutiger symbolischer Aufgabenplanung und Motoransteuerung. Denn gerade heutige Roboter müssen sich in komplexen und sich ändernden Umgebungen zurechtfinden können. Damit sie das tun können, muss eine Verarbeitung von physikalischem Allgemeinwissen möglich sein. Die physikalische Vorhersage von Roboteraktionen ist dabei die Paradedisziplin von Physiksimulationen, wie Gazebo.

In der fachverwandten Literatur werden bereits symbolische Planung und Schlussfolgerung mit geometrischen sowie zeitlichen Komponenten, oder der Aufzeichung von offline simulierten Aktionen, erweitert.

Das Ziel dieser Arbeit ist die Beantwortung der Frage, wie ein planungs-basiertes Robotersystem sein Verhalten durch physik-basierte Vorhersage von Roboteraktionen verbessern kann. Der zielgerichtete Ansatz ist hier die Integration einer physikalisch realistischen Vorhersage um Ausführungsfehler zu verhindern. Erreicht wird das durch die Anpassung oder Veränderung des aktuellen Roboterplans und seiner Aktionen. Die Arbeitshypothese lautet deshalb, dass solch ein integriertes Vorhersagesystem Pläne effizienter ausführt als ein System ohne dieses.

Im Verlauf dieser Arbeit wurden eine probabilistische Vorhersagemethode, “Functional Imagination”, und eine Systemarchitektur entwickelt. Für die praktische Evaluation wurde der PR2 Roboter und Gazebo benutzt. Die Vorhersagemethode wurde in ein Blackboard-basiertes Roboter-Kontroll-System integriert. Experimente verifizierten die Simulation als genau und sehr nahe an der Realität. Die Ergebnisse von drei experimentellen Szenarien in einer Restaurantdomäne unterstützen die Hypothese, dass die physikalische Vorhersage in Kombination mit Planungs- und Ausführungs-Komponenten einem System ohne diese Vorhersage überlegen ist, hinsichtlich Robustheit und Effizienz.

Am Ende werden die Ergebnisse, Einschränkungen und mögliche zukünftige Forschungsfelder aufgezeigt und ein kurzer Ausblick im Simulationskontext gegeben, wie verteiltes Rechnen und Cloud-Computing das Forschungsfeld der Robotik in Zukunft beeinflussen werden.

Acknowledgements

In working on this dissertation, I have greatly benefited from the help and support of many people. Here are those I would like to thank especially.

First and foremost, I want to thank my advisors, Jianwei Zhang and Bernd Neumann, for their confidence in me realizing this work and their valuable advice. I would also like to thank William Morris for taking the time to read and comment on many drafts of this dissertation. Furthermore I am particular grateful to Denis Klimentjew for his hints and comments during my study and for becoming a friend.

Carrying out successful research in autonomous robotics is not possible without being a member of a team. I had the luck of joining two of the best teams. Firstly the RACE team: I would like to thank especially Joachim Hertzberg for very encouraging discussions, ideas and comments on many stages of this work. Secondly special thanks goes to the members of the TAMS team for providing me with one of the best technical environments and expertise, for working with autonomous robots. I owe especially many thanks to Eugen Richter for his time and help, often on very short notice.

Most importantly, I want to thank Ricarda for her love and support, helping to create this manuscript.

Table of Contents

Introduction

1

1.1 Motivation and Goals

From the early days of robotics, a fundamental and interesting question has always been how robots could adopt human cognitive processes. In this work an approach is proposed that implements the principle of cognitive imagination, otherwise named *functional imagination* [MHN08]:

> We define functional imagination in the context of artificial embodied agents as the mechanism that allows an agent to simulate its own behaviours, predict their sensory-based consequences and extract behavioural benefit from doing so. By behavioural benefit we mean an increase in reward or utility achieved by using internal simulation.

The approach combines into one hyperreal system the cognitive concept of imagination with high-level planning and execution. The proposed Hyperreality Imagination-based Reasoning and Evaluation System (HIRES) does not rely upon traditional spatial reasoning [RN07], but instead uses simulation as a tool that enables the testing of robot execution before actions are actually performed. This naturally saves robot resources and, when optimized, total execution time, as well as improving system robustness. The approach is inspired by the concept of imagination [Cas71]. Imagination enables humans to consult rules or principles but not merely to apply those rules [Loa01]. Instead humans imagine what the consequences might be if the rules are, or are not, followed [Lod96]. The author notes that humans constantly use imaginative projection and that imagination is essential for human reasoning. So why should it not also be used for robot reasoning in a human-robot environment?

According to Loasby [Loa01], reasoning is expensive in time and energy and should therefore be used sparingly. The approach of this thesis is to combine the concept of imagination with the concept of 'Hyperreality' [TT01]. The term Hyperreality comes from earlier work by Baudrillard [Bau94]. It originates in semiotics and postmodern philosophy and is considered to be the condition in which an agent cannot distinguish reality from a simulation of reality. Therefore reality and virtual reality can be considered interchangeable at some point. This blending of virtuality and reality can be seen

in a broader sense as mixing human and artificial intelligence[1]. Baudrillard postulates that with 'Hyperreality' there is no longer a difference between real and imaginary - both exist at the same time. This seamless transition between simulation and reality is related to the term Mixed Reality (MR) as employed in [RKZ12, CMW09]).

An exact simulation of a scene can be used in order to execute robot actions (e.g. picking-up a mug from a counter and placing it in front of a guest) before they are actually performed. The consequence of simulating the action might be that the current plan must be adapted in order for the task to succeed (e.g. after the failure of placing a mug in simulation).

The results from this work support the idea that applying a functional method of imagination within the context of a hyperreal system improves robot autonomy and saves resources. Results are described in Chapter 6.

1.2 Motivating Scenarios

To illustrate and evaluate the performance of such a system, a real-world dynamic environment should be appropriate. Thus the evaluation of this work employs various scenarios in which a mobile service robot performs tasks in a human environment. These scenarios demonstrate how well the approach predicts spatial, temporal and physics-related action results. As these scenarios benchmark the whole system, they are also a measure of how well the system components are integrated. In the following paragraphs, some possible situations are described.

The experimental domain is a restaurant-like environment in which the robot carries out tasks typical for a waiter, such as serving a beverage to a guest or cleaning a table. Aspects of these tasks require reasoning in order to be performed correctly. For example, to place a mug on the table near to the guest, the robot must use spatial reasoning to compute a suitable table-top area in which the mug is to be placed. The appropriate area might already be occupied by the guest's belongings, e.g. a mobile phone or a book, or by normal table decorations such as a vase or salt and pepper pots. Considering this scene in a restaurant with a spatial reasoning procedure causes an increase in "reasoning space" as more objects are considered, because computations have to consider each object in relation to all other objects.

One can imagine a situation in which the robot must place a cup of coffee on a restaurant table with a guest waiting. Typical constraints on a waiter include placing the mug in front of the guest, often with the handle directed towards the right hand for convenience. State-of-the-art mobile robot manipulation capabilities allow the table to be found. Prior knowledge might allow the robot to query the best place for the mug.

[1] Baudrillard rejects the reference to his work in the movie trilogy "The Matrix" where *Morpheus* refers to world outside the Matrix as the "desert of the real".

Nevertheless the table might be occupied by other objects such as plates, cutlery or the guest's belongings, such as a mobile phone. Modern reasoning and learning systems consist not only of knowledge representation systems and planning modules, but also contain reasoning and planning components. A spatial reasoning unit is concerned with the spatial relation between objects (on the table) and calculates the best place to put the mug, whereas a temporal unit tells the robot to deliver the drink while it is still hot. A hierarchical planning module coordinates robot capabilities at a symbolic level. Such modular task sharing increases the autonomous robustness of intelligent mobile systems and represents the state-of-the-art.

One can still imagine a situation in which the robot is serving a drink and the spatial planner finds that no area of the table is free. This would cause the 'place' action in a planning chain to fail and thus would either cause the whole plan to fail or would trigger replanning, if possible. Whereas a Hierarchical Task Network (HTN) planner, or similar, operates at a purely symbolic level, the spatial reasoning module typically considers the bounding boxes of objects and therefore considers more geometric details. In this work the spatial reasoner considers the objects on the table represented only as two dimensional (2D) objects. So what happens when an area of the table is unoccupied but that area is not reachable by the robot due to kinematic or collision constraints relating to robot geometry? Or what happens when the spatial reasoning module finds no free space, e.g. because although there is space available the placement tolerances are too high? A more sophisticated example can be imagined in which objects may be stacked but the mug should not be placed on a sloping surface. In a simulation, these situations can be reconstructed by applying the placing routine multiple times with different parametrization. This allows the feasibility of such a parametrized placement to be considered.

In another possible scenario, the robot might fetch an object from a counter and carry it on its tray. Planning and hybrid reasoning is based on the input and output of a so-called "carry-object" task. The system can only react to the outcome - either fail or replan if possible with another object of the same kind. Physical parametrizations, such as trajectory, velocity and acceleration, are typically the domain of a physics simulation or a motion controller. Here the carrying of an object can be accurately simulated in order to predict the toppling of the object on the tray over a range of robot motion parameter variations.

As described earlier, temporal constraints ensure, for example, that a cup of coffee is still hot when placed in front of a guest. The basic calculation here is limited to either hand-coded or learned knowledge about general events for a robot moving in a restaurant. In a simulation, it is feasible to measure the exact time of a given trajectory by letting the robot execute it. Accurate timing can then be the basis for reasoning about which actions the robot should take.

The two motivating examples above need good parameter values in order to succeed. Finding these values with a physical simulation needs multiple simulation runs, as there

is usually not just one best parameter (to test). Even though certain parameter values have a high likelihood of success in simulation, the outcome of the operation might still be different in reality. Hence this uncertainty has to be dealt with appropriately.

1.3 The Faculty of Imagination

Imagination is based upon images in the human mind that are not perceived through the ordinary senses. When somebody imagines the color of the sky at sun set, this might involve 2D pictures. Imagining the rotation of a cube in order to 'see' the rear face might involve three dimensional (3D) images. In principle it should be possible to create these images in a robot's 'mind'.

In this work the term 'imagination' is not related to the technical process in psychology of reviving in the mind aspects of objects formerly perceived. This rather 'reproductive' process conflicts with the 'productive' process of imagination.

As the robot's 'mind' is not technically defined, imagination must be implemented as a function within a robot control program. More concretely, a 3D physical simulation can render 2D images with colors and textures in a very detailed manner close to reality as well as creating 3D motions with physical embodiments. Thus there must be a deeper reason for the fact that cognitive scientists see a fundamental difference between imagination and physical (or visual) simulation.

The answer might be found in the work of Ned Block and Philip N. Johnson-Laird. Block, the author of [Blo81], describes the long-lasting and still current "great debate" on the nature of mental images: are images pictures in the head or are they like the symbol structures in computers? In other words, one could ask whether mental images are the consequence of cognitive processes or the tools of those processes. In another work the author claims that mental images do indeed exist, both 2D and 3D [Blo83]. Are people *rotating* mental images at measurable speeds? Imagine an experiment in which the task is to look at pairs of pictures showing rotated 3D blocks and to decide from the pictures whether or not the blocks are congruent. The experiment shows that it is possible to measure the speed at which images are rotated in the human mind. Further experiments are presented, for example *Map Scanning* and an explanation of *Mental Image* is given.

The author of [JL83] deals with the fundamental question of the nature of mental models. He argues that humans understand the world by building inner mental replicas of the relations between objects and events. The mind is therefore a model-building device that can itself be modeled on a computer. Furthermore this work provides a guide to the building of such a model. In [JL95], the same author considers the two main approaches to deductive thinking: formal rule-based inference as a syntactic process; and mental model theory, which includes the idea that deductive thinking

is a semantic process. The conclusion is that "Models are the natural way in which the human mind constructs reality, conceives alternative realities and searches out the consequences of assumptions." Models are the medium of thought, proposed [Cra43]. According to Craik,

> "thought is a term for the conscious working of a highly complex machine, built of parts having dimensions where the classical laws of mechanics are still very nearly true, and having dimensions where space is, to all intents and purposes, Euclidean. This mechanism has the power to represent, or parallel, certain phenomena in the external world as a calculating machine will parallel — and so be able to 'give an account of' or 'predict' most easily — those phenomena whose mechanism most resembles itself" (pp. 94-95).

But what *is* a mental model? Craik described the brain's internal models as follows:

> "My hypothesis then is that thought models, or parallels, reality — that its essential feature is not 'the mind', 'the self', 'sense data', nor propositions but symbolism, and that this symbolism is largely of the same kind as that which is familiar to us in mechanical devices which aid thought and calculation".

Craik goes further as he considers the use of physical models (for him they can serve also as symbols), as well as words and numbers for prediction and reasoning. Furthermore he states that a mental model does not have to resemble the object or situation pictorially "since the physical object is 'translated' into a working model which gives a prediction...". In contrast to defining mental models, this thesis is about finding appropriate functional models that are computationally representable.

1.4 Research Question and Contribution

In this work one fundamental question posed is whether and how a physical simulation can be used as a companion tool for predicting the results of robot actions. Another question is how this can improve robot competence, both quantitatively and qualitatively. Competence here is referred to as adapting behavior in a goal-directed way to save resources and succeed with the task at hand.

In order to verify that simulation is an appropriate tool for prediction, it has to be evaluated with respect to its quantitative similarity with the real world. This is discussed in Chapter 6. Therefore reasonable metrics have to be proposed, applied and evaluated. Based on the assumption that the simulation in use has a sufficient degree of realism, a prototype system able to exploit simulation as a companion tool must be integrated. The evaluation is based on the scenarios and software developed in the Robustness by Autonomous Competence Enhancement (RACE) project in the context

of the European Union (EU) Framework Programme for Research and Technological Development (FP7). The RACE project is described in Chapter 4 and the scenarios are described in Chapter 6. In order to support the hypothesis posed below, an evaluation metric benchmarking the robot's competence has to be defined. This is done for each scenario and is described with the related experiments in Chapter 6.

The research question, approach and hypothesis of this thesis can be refined as:

Research question: *How can a task planning based robot system be improved by prediction derived from physical simulation?*

Approach: *Integrate a realistic prediction of robot activities to allow these activities to be adapted (by a plan change or by replanning) before a sub-optimal (failing) activity is executed.*

Hypothesis: *A system integrating such prediction is more efficient (such as requiring less time, plan steps) than a comparable system without.*

Answering the research question and verifying the stated hypothesis in this dissertation spans multiple topics:

- Design & Integration: How to integrate the cognitive science principle of 'imagination' into a mobile service robot.
- Physics & Robot Motor Control: Is an off-the-shelf physical simulation accurate enough to predict robot action results?
- Task planning & Execution: Does an integrated imagination approach improve plan-based execution?
- Algorithms & Data structures: Which algorithms and representations have to be implemented to handle and reason about simulated robot action results?
- Motion Control and Navigation: How can robot motion control and navigation be reasonably parametrized?

1.5 Structure of the Thesis and Related Publications

This thesis is organized as follows:

- Chapter 2 gives an overview of related literature in relevant fields.
- Chapter 3 describes in detail the hardware platform, sensors and actuators used.
- Chapter 4 gives an overview of the software tools and frameworks used for this work and discusses the decisions that led to the approach taken in this work. Contributing publications were [HZZ+14, RNZ+13].

- Chapter 5 contains the main contributions of the work. After stating the objectives, a methodology and system architecture that integrate functional imagination are developed. The proposed system and its use cases are described. Publications that contributed to the chapter were [RKZZ14, CNSV14, KRZ13, ZRZ13, EKRZ13, KRZ12, RKZ12].
- Chapter 6 describes the experiments conducted and the way in which results were collected. It also evaluates the results. [ZRP+13] contributed to this chapter.
- Chapter 7 concludes this thesis and gives an outlook of future research.

Work during the years leading up to this thesis has resulted in several publications that were accepted at conferences, workshops and seminars, including: contributions relating primarily to plan-based, reasoning-and-learning mobile service robotic systems and their software architecture; simulation-based reasoning in mixed-reality environments; co-operative multi-robot systems; robot action prediction and system evaluation. Appendix A.9 gives a comprehensive list of all publications resulting from this work.

Related Work

2

This chapter presents a summary of related work, starting by describing current approaches to symbolic planning and recent extensions, such as geometric, spatial and temporal planning. The chapter also introduces the main topic of this work - the combination of simulation with planning and execution - and describes how this approach goes beyond related work. It also discusses some approaches to the evaluation of intelligent robotic systems and introduces several software frameworks used partially in this work. Various technical terms will also be introduced.

2.1 Symbolic and Geometric Task Planning

State-of-the-art reasoning and planning is based on symbolic, hybrid (spatial, temporal, causal), kinematic and collision information, as well as on parameters obtained through theoretical deduction or simulation. A classical approach to task planning is the STanford Research Institute Problem Solver (STRIPS) method [FN71]. STRIPS-based planners focus on achieving a specified goal state, whereas HTN planners focus on solving abstract tasks. The latter is generally more efficient than classical planning because the search is limited to a given library of recipes. In this work a popular HTN planning method called "totally-ordered" HTN planning with partially ordered sub tasks (henceforth simply called HTN planning [NMAC+01]) is used. In this hierarchical approach, methods and operators are modelled along with their pre- and post-conditions (effects). Decomposition of the problem definition creates a hierarchy of methods and operators to solve the task at hand, as shown in Fig. 2.1. As an optimization, in this work the resulting plan is scheduled for parallel execution if the robot has the necessary resources. This is described in [EKRZ13].

Extensions to traditional symbolic planning techniques incorporate additional constraint processing techniques. They are based on a temporal knowledge representation (KR), such as Allen's Interval Algebra [All84]. The Meta-Constraint Satisfaction Problems (Meta-CSP) framework[1] [MP06] enriches these representations with metric constraints [vB90], see also Fig. 2.2. This method searches in a tree-like structure for

[1] http://metacsp.org (March 16, 2015)

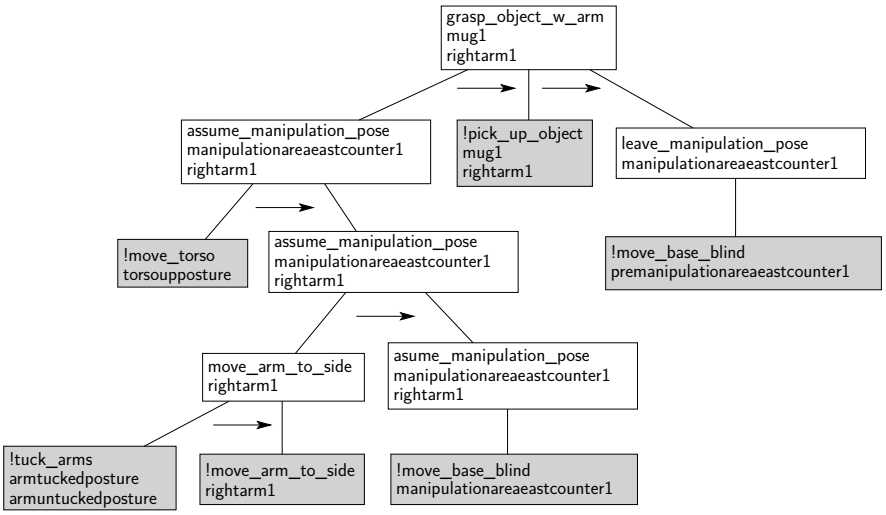


Fig. 2.1: Hierarchy of HTN methods: grasp a mug with the right arm. Primitive operators are in grey boxes and arrows indicate method decomposition (from left to right). [Courtesy of Sebastian Stock, University of Osnabrück]

a solution to the problem at hand and back-tracks if needed. This hybrid reasoning and planning method combines spatial, resource, action and ontological models on symbolic and geometric levels.

[dSPA13] combines a geometric task planner (GTP) with a symbolic HTN planner employing backtracking (Fig. 2.3). Certain actions and operators are of a geometric nature and so are checked by the geometric planner ("lazy checking"). Checking happens during planning, before the creation and execution of the final plan. Predicates like `isReachable(Object,Human)` are checked and are preconditions for HTN planning methods. Evaluation focuses on the system execution time and especially the run-time overhead of the GTP. Recent work [dSGPA14] uses interleaved backtracking to (try to) close the gap between symbolic and geometric planning.

[LB13] proposes a hybrid reasoning approach to autonomous mobile manipulation systems. Symbolic planning is integrated with geometric reasoning to fulfill a mobile pick-and-place task. Actions are simulated step-wise until the goal state is reached or otherwise they are back-tracked. Evaluation uses the proposed system for reachability analysis while taking two mugs from a table to a shelf.

[DHN13] presents an integrated task and motion planning system (evaluated on the Personal Robot 2 (PR2)). It combines symbolic and geometric reasoning. The authors apply a continual planning system in which replanning is automatically triggered during run-time. Geometric reasoning is implemented as an external function call built into the precondition of the (planner) task as a "semantic attachment" delivering geometric facts. An additional condition checker module determines the truth values of

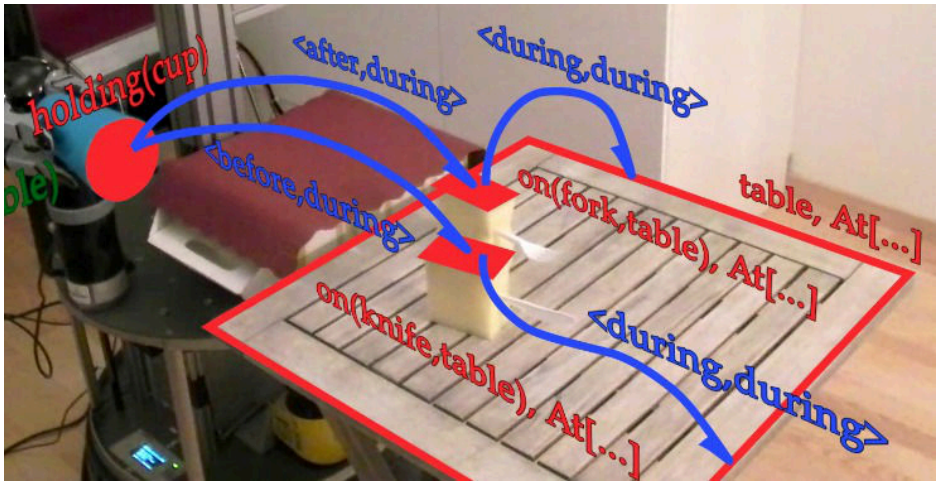


Fig. 2.2: Planning with temporal and spatial constraints [Courtesy of Iran Mansouri, Örebro University]

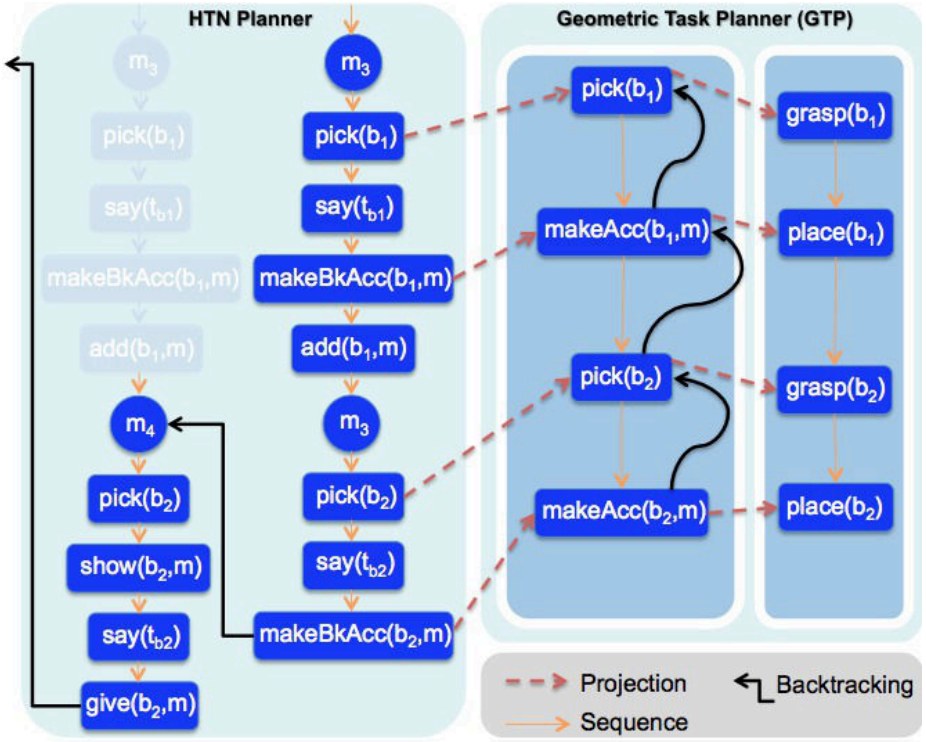


Fig. 2.3: Planning with geometric constraints. A symbolic planner is combined with a geometric planner enabling projection and backtracking. [dSGPA14]

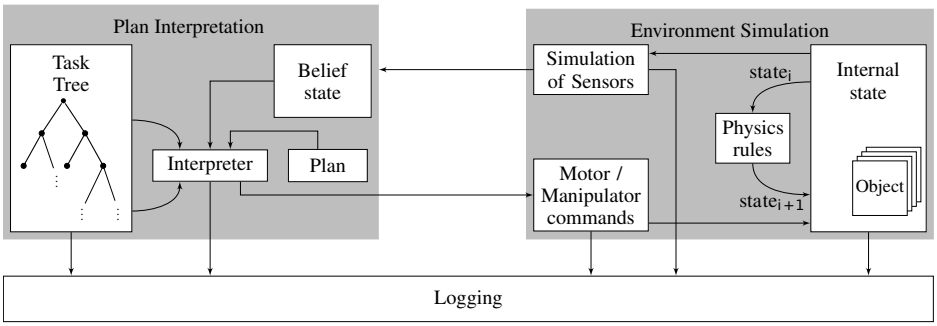


Fig. 2.4: Temporal projection using simulation. [MB09]

logical variables by an external function call, e.g. `canPutdown` (while placing an object). A cost module computes operator costs using an external function. The authors observed an increase in planning and execution time as more objects were present. Another target of the paper was to test the impact of lazy evaluation and cached searches on overall performance. Evaluation focuses on measuring the relative planning times of 'lazy' vs. 'eager' approaches. Further evaluation aspects involve execution times and (table) actions, cache misses and subsumption.

[MB09] investigates how a database of projected robot behavior in household tasks can help a planning system prevent costly misbehavior. Information is logged and translated into first-order representations of robot plans. This allows the planner to infer world states at (sampled) projected times and thus to recognize various plan failures in manipulation tasks. The work concludes that many robot-behavior failure conditions that are difficult to represent in symbolic transition modes can be recognized at very low cost in physics-based and sensor-based simulations. Nevertheless, according to the authors, "[...] simulators, do very little abstraction, time has a high resolution, [...] However, the realism and accuracy comes at a price. Simulations can only sample possible episodes but not quantify over all possible ones." The essence of the work seems to be to build a repertoire of projected robot actions from which the planner can choose. It does not try to build a system to project behavior on-the-fly based on current world state, i.e. the state perceived by the real robot sensors. Fig. 2.4 shows the connection between simulation and interpretation in the authors' approach.

In contrast to a general action approach, the authors of [ZV10] developed Variable Level-Of-Detail (VLOD) motion planning, employing a simulated physics-engine, the *NVIDIA PhysX*. This is based on Rapidly-exploring Random Trees (RRT). The goal was to enable efficient replanning in changing environments with the help of a physical simulation.

[DHCS12] presents a planning method for operating in cluttered environments. The author's method is a physics-based object-centric approach with a clutter-centric perspective. To predict the outcome of an intended grasp, the authors use a physics-based

simulation method that computes the reaction of objects in the scene to a set of possible robot movements. It considers simultaneous object interactions that can be pre-computed and cached. The authors state that object-object interaction would require run-time capability, which is left out. The approach allows more possible grasps to be found for a given scene and it succeeds in more scenes. Validation uses a PR2 showing that the predicted outcomes match real-life results. It is a partially online approach that also explores sampling as the method of choice when handling uncertain parameters. A current limitation is that it considers only 2D collisions, i.e. no toppling or stacking of objects.

[BHKLP12, BKLP13] defines the diverse action manipulation (DAMA) problem concerning a robot, a set of movable objects and a set of diverse manipulation actions. The aim is to find a sequence of actions that moves each of the objects to a goal configuration. The problem is addressed as a multi-modal planning problem and a hierarchical algorithm is proposed to deal with it. Although the work addresses robot manipulation problems that are similar to those addressed by this thesis, the authors do not use physics planning or simulation to improve the robot's autonomy or robustness.

[KLP12] presents geometric and symbolic representations in a hierarchical planning architecture called Hierarchical Planning in the Now (HPN). A system is integrated to generate behavior in a real robot that robustly achieves tasks in uncertain domains. Simulation is used to demonstrate the system but physics capabilities are not exploited for reasoning.

[LDS+14] combines symbolic and geometric reasoning. The authors use the Planning Domain Definition Language (PDDL) and compute geometric alternatives for whole-body control with null space projections.

[KLP+14] uses an online Google text corpus to extract commonsense rules for use in robot planning.

HTN planning is currently one of the most popular high-level planning strategies for mobile robotics. In this work the Simple Hierarchical Ordered Planner 2 (SHOP2) [NMAC+01] is used. A huge advantage of SHOP2 over other HTN planners is its ability to generate partial-order plans. Unfortunately, HTN planners provide no facilities for parallelization or replanning. To compensate, in this work the planner is extended with a parallelization layer realized with State MACHine (SMACH) [BRGJ+11]. This allows the use of containers to assign different tasks to each state and thereby enables any type of parallelized plan to be executed. This early work is described in [EKRZ13].

[KLP12] combines geometric and symbolic representations into a hierarchical planning architecture, referred to as HPN. A system is integrated to generate behavior in a real robot that robustly achieves tasks in uncertain domains. Simulation is used to demonstrate the system but not to exploit physical reasoning capabilities. [BHKLP12, BKLP13] define the DAMA problem concerning a robot, a set of movable objects and a diverse set of manipulation actions. The objective is to find a sequence of actions

that moves each of the objects to a goal configuration. The manipulation problem is addressed as a multi-modal planning problem and a hierarchical algorithm is proposed to deal with it. Although the work addresses robot manipulation activities similar to those in this thesis, the authors do not use physical planning or simulation to improve robot autonomy or robustness.

Most state-of-the-art planning approaches are based on symbolic information (i.e. abstract information) and lack commonsense reasoning. Although current methods are promising in combining high-level and low-level knowledge representation with reasoning and planning, there is still a gap. This is especially true when working in changing and partially unknown environments where object information such as size, weight, and inertia are not even modelled.

2.2 Projection, Simulation, Imagination and Prediction

In [KDB11] a system projecting robot action results in everyday object manipulation tasks (making a pancake in this case) is proposed in order to gain appropriate action parametrizations for successful execution. This complicated task (for a robot) is broken down into simpler actions, such as breaking an egg, pouring the pancake mix (Fig. 2.5), etc. A model integrated with PROLOG allows chronological backtracking of actions and triggers physical simulations, monitoring and logging. It translates the logged data into first-order time-interval-based representations, called timelines. Naive physics and commonsense reasoning are used to improve the robot's manipulation capabilities. The framework includes the Gazebo simulator, the PROLOG logic programming language and a parametrizable robot control program. Evaluation focuses on qualitative results (e.g. a broken egg) related to specific parameters. The authors state that today's methods of symbolic reasoning that infer the behaviour of simple physical problems are of limited use because the necessary detailed physical properties are abstracted away. The system presented uses a high-fidelity physics model that lacks real-time performance and must therefore be applied offline (a-priori). Predictions cannot be made quickly enough to be useful for planning during execution. The work does not use higher level symbolic planning (e.g. HTN planners) or state machines (e.g. SMACH). [KDGB11] focuses on the combination of first-order symbolic representation with physics-based simulation as an inference mechanism for predicting the effects of actions. This allows for probabilistic predictions. [KHB12] uses a data glove to interact virtually and to demonstrate activities, such as pouring a pancake mix, and translates the logged data structures into first-order representations (timelines). This is planned for use with a robot performing similar tasks. Parametrization takes place by iteratively simulating changing robot action parameters and evaluating the results. Once again, simulation is used offline.

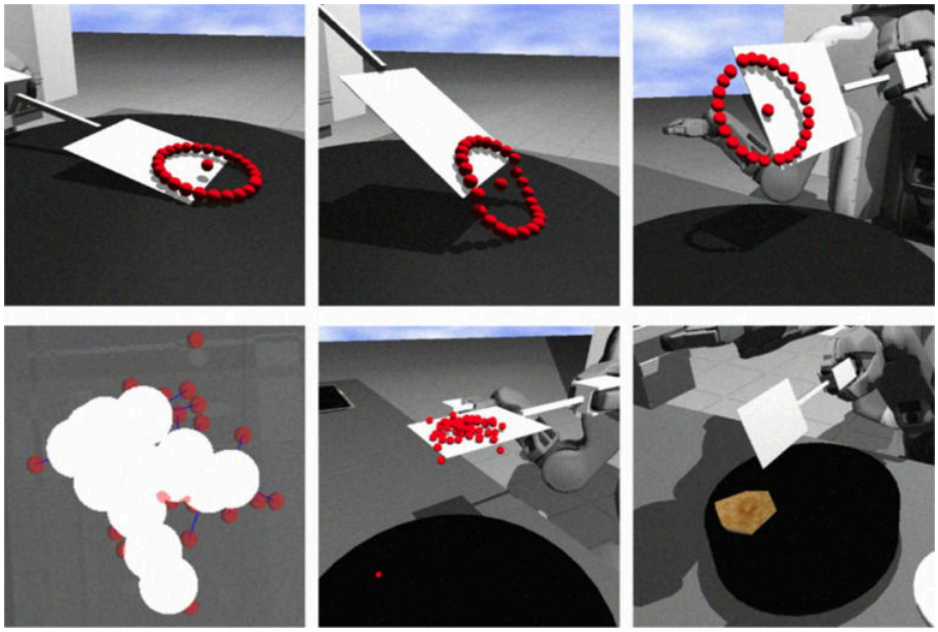


Fig. 2.5: Different actions in a pancake making scenario: typical action, such as flipping the pancake, are tested in simulation in order to find good action parameters, e.g. the angle by which the wrist must turn. [KDB11]

[BB07] presents a qualitative evaluation of a number of free, publicly available, physics engines for simulation systems and game development. Simulation can be described as solving the forward dynamics problem: given the forces acting on a system, how does the system move as a result? The authors present six major metrics of simulation performance: the simulator paradigm, the integrator, object representation, collision detection, material properties and constraint implementation. Of the ten physics engines evaluated by [BB07], this work mainly uses the Open Dynamics Engine (ODE)[2], version 1.0 of which is used in the Gazebo simulator. Bullet[3] is compared as an engine supported by recent versions of Gazebo.

In [BHH04] a mobile robot control architecture is introduced for projection-based planning, execution and learning. It supports the projection of navigation plans based on learned models of such actions. This information is used to improve the robot's navigation plan. The results show that navigational models learned in simulation can improve physical robot performance.

In [AKPL13], the authors show how to use simulation to detect faults caused by the environment when releasing an object. The results are used to modify actions in order to avoid such faults. The theory of these so-called external faults is investigated frequently. Such faults are sometimes also called *interaction faults* or *unfavorable*

[2] http://www.ode.org (March 16, 2015)
[3] http://bulletphysics.org/wordpress (March 16, 2015)

environment conditions and are also labeled as *unforeseen events*, *exogenous events* and *errors*. The authors use the ODE physics engine to predict robot action results. They evaluate their approach by comparing their N-Bins algorithm to popular machine learning (ML) algorithms, such as decision trees and k-nearest neighbors. Results are judged by comparing the model properties (such as object kinematics and dynamics) of the initial and final states. The approach is evaluated on a 'release' action of a robot manipulating an object. Simulation was used to estimate a good release state for the manipulator in order to prevent external faults. Although this approach limits the simulation use-case to one action, it applies a similar paradigm to the prediction of robot action results: simulation, evaluation and action modification.

In [JLS12] the authors allow a robot to arrange the human environment by letting the robot anticipate and imagine human positions and then figuring out what those virtual humans are likely to do. This approach relies on learning how objects relate to human poses based on their affordances[4], ease of use and reach-ability.

[MKNH08] defines the term *functional imagination* as the purposeful manipulation of information that is not directly available to the senses. It states that imagination always points to something that in reality is not there and that in order to be useful, imagination also needs to provide information about the likely consequences of actions. This is then to be used by a mechanism for translating imagined motor actions into sensory-based representations of their consequences. In [MHN08] the authors propose a framework for modelling functional imagination: an embodied agent that simulates its own behaviors, predicts their sensory-based consequences and extracts behavioral benefit from doing so. The system is based on a physics-based humanoid simulator - SIMNOS [GNHK06].

In [KS13] the authors developed a system that anticipates what a human will do next with the aim of allowing an assistive robot to plan ahead in response. A temporal, conditional, random field represents each possible future.

The term *robot imagination* was used in [VMJB13] for a system that generates models of objects prior to their perception. Robot imagination is defined here as the robot's capability to generate feature parameter values of unknown objects by generalizing characteristics from previously presented objects.

Simulation has become an essential tool in the development of complex mobile robotic systems. The availability of physical robots is limited and researchers often work in separate physical locations on different system parts. Consequently whole-system evaluation and testing is often not possible. In such cases a realistic simulation is indispensable. Simulation as a verification and testing tool is widely adopted during

[4] The term 'affordance' was coined by psychologist J.J. Gibbson (1904-79). In Gibson's formulation, affordances are properties of the environment, independent of the perceiver. Later, Don Normen, a student of Gibbson, replaced the term with the notion of the actual and fundamental properties of the thing, such as how it can be used [Nor02].

system development. Recent research into autonomous robot control extends the use of simulation for reasoning.

Harris et-al. [HC11] give a good outline of the available simulators, frameworks and tool-kits. The authors suggest that most simulators are not well documented (for anything but Windows environments), have no sensors or offer no possibilities for the integration of new robots and sensors. Many tools are insufficiently precise, do not include physics engines or are no longer under active development. However, according to the authors, Robot Operating System (ROS) has been proven to be very suitable. The Gazebo-based simulator provided by ROS integrates many robots and sensors as well as many libraries and tools. It also provides a physics engine and delivers accurate results. Furthermore ROS is designed to be a partially real-time system according to Harris. This, in addition to very good PR2 integration, explains why ROS is currently one of the most popular robotics frameworks.

[KyHR03] developed a robotic system that incorporates a coupling between simulation and perception that keeps an internal world model of the robot's environment up-to-date. This so called "object permanence" is used for conversational robots that can ground the semantics of spoken language with sensorimotor representations. The motivation for the work comes from the extensive use of visualization and imagination in human cognition.

[LC01] discusses how a robot agent appears to be controlled by a human and also discusses artificial intelligence in a broader sense. It defines requirements orthogonal to traditional artificial intelligence (AI) topics (such as planning, vision and learning) - namely being animate, adaptable and accessible. Such a robotic system is called a semi-sentient robot.

[KGT01] is a philosophical and psychological view of mental imagery. It discusses mental imagery as a basic process of cognitive functions, tackling interesting questions about the mechanisms used in other activities, such as perception and motor control. The work concludes that imagination affects the body, much as actual perceptual experience can.

[Tho14] discusses *Mental Imagery* historically. According to the authors it is referred to as 'visualizing', "seeing in the mind's eye", "hearing in the head", "imagining the feel of," etc. It is a quasi-perceptual experience that occurs in the absence of appropriate external stimuli. Is it a mental representation or an experience? Despite this long-lasting discussion, it is believed to play a large role in both memory and motivation, in visuo-spatial reasoning and in inventive or creative thought. The work states that *imagery* is "a dynamic process more like *active perception* than a passive recorder of experience". Technically it mentions a development towards modeling mental images as *digitized* pictures generated within a computer simulation. It has also influenced cognitive science.

As already mentioned in Chapter 1, handling uncertainty is essential when working with prediction models in the real world. Models are used to represent the probability that a certain robot action succeeds. In addition they can determine suitable robot action parametrization.

Beetz' work [MB09, KDB11, KDGB11] with Bayesian networks seems suited to the calculation and representation of the probabilities of action results. Combining the probabilities, here called 'nodes', gives the success probability of a whole robot plan. The author considers nodes to be independent. Furthermore the latter approach handles uncertainties (caused by sensor noise) during simulated execution by projecting the same plans multiple times. As projection takes place offline the approach developed handles uncertainty by sampling single simulations per parametrization online.

In order to represent parameter sets, e.g. for motion parameters, a defined or learned fuzzy set could be appropriate. Furthermore to learn from and reason about the relations between motion parameters and toppling events, a representation for speed parameters might have to be defined, e.g. a unifying vector for angular and linear sets.

Current HTN planners take only success and failure into account, not the probabilities of action results. A method of representing and handling probabilities has to be defined.

In [Bee00] the author uses a weaker form of imagination integrated into (re-) planning and execution monitoring employing a dedicated planning language. However plan generation does not scale well due to the complexity of the problem.

Predicting the results of robot actions is an interesting field of research. Two approaches for arm motion and motion parameter prediction for a mobile base are briefly discussed below. These methods do not, however, use simulation to derive prediction.

[BBF14] deals with "Learning Predictive Models of a Depth Camera and Manipulator from Raw Execution Traces". The requirement of this approach is to show that a Hilbert Space Embeddings variant of a Predictive State Representation (PSR) can learn a model of a Color and Depth (RGB-D) camera and robotic arm directly from uninterpreted actions and observations. The overall goal is to predict future actions. The approach represents an arm state as a set of conditional distributions of future observations given future and historic actions. The results show that it is possible to predict the movements of a 7-degree-of-freedom (DOF) arm after training.

[OSB14] developed a learning-based nonlinear model for predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. The system is able to predict the right motion parameters depending on the surface (recognized visually).

On the basis of an accurate and recent simulation, the border between simulation (or virtual reality) and reality fades away. In philosophy and sociology the term *Hyperreality* has been established to describe exactly this condition [Bau94].

Lodato [Lod96] describes imagination as being central to human reasoning in that it allows us to organize and describe our experiences. And Loasby states that the creation of new patterns rests on imagination, not logic. In this work, robot imagination enables reasoning about possible robot actions by inferring from simulation.

2.3 State-of-the-art Robot Operating Systems

A Multi-Robot System (MRS) allows multiple robots to operate in parallel. More importantly, it supports the running of encapsulated system components in a distributed manner. In the following section the two main open source multi robot systems are briefly introduced.

The Player Project, described in [GVH03], was previously used widely. It is a software MRS providing tools that simplify controller development, particularly for multiple-robot, distributed-robot and sensor network systems. Typically various heterogeneous mobile robots are controlled by a dedicated MRS.

An MRS provides hardware abstraction and driver encapsulation, where a driver is a control algorithm that supports the underlying hardware. For example, a driver might control differential drives, ranging sensors or localization. Furthermore an MRS typically provides tools, drivers and software frameworks for accessing a range of different robots. Generally, concurrent robot activity and inter-robot communication are supported. Nevertheless there is currently no framework for intelligent, networked device behavior.

The Player Project offers a basic set of functionality for every robot: perception, manipulation and representation of the environment in the form of a map. In particular, any client based on this MRS can rely upon motion control, object manipulation (such as controlling a robot arm), perception of the environment via sensors and motion planning towards a specified target. Motion planning includes the mapping of sensory data to map coordinates (localization) and navigation.

A successor to the Player Project is ROS, which was based on central Player components, such as the navigation implementation. It adds state-of-the-art functionality, such as a publish-subscriber mechanism, to nodes.

ROS[5] provides features similar to those of the Player/Stage project and uses algorithms from that project and others [QCG+09]. Its focus is more on dynamic, peer-to-peer communication between robot devices and on modular design. ROS supports client

[5] http://www.ros.org (March 16, 2015)

code in C++, Python, Octave and LISP. Other language support, such as a Java-Application Programming Interface (API) via Java Native Interface (JNI)[6], is either planned or currently in an early phase of support[7].

ROS is an open source project consisting of many software modules that support a variety of users and developers.

In this thesis, ROS is used as a low-level framework and its integrated simulator proves to be suitable for research, testing and evaluation.

2.4 Benchmarking and Evaluation of Intelligent Robots

The evaluation of intelligent autonomous systems has been explored widely in the literature, although 'intelligent' has no clear definition. Very little work has concentrated on improving robot competence by learning from mistakes. One example of such work is a system developed as part of an EU project that targets learning from experience, described in [RNZ+13]. Some approaches for evaluating intelligent systems are presented below.

The RoboCup@Home league[8] intends to develop service and assistive robot technology for domestic applications. The main topics concerned are human-robot interaction and cooperation, navigation and mapping in dynamic environments, computer vision and object recognition in natural light conditions, object manipulation, adaptive behaviors, behavior integration, ambient intelligence, standardization and system integration. The only requirement for participation is the provision of an autonomous robot that can take part in at least one real-world scenario. At the time of writing, the test criteria include that the project should:

- include a human-machine interaction,
- be socially relevant,
- be application directed/oriented,
- be scientifically challenging,
- be easy to set up and low cost,
- be simple and have self-explanatory rules,
- be interesting to watch,
- take a small amount of time.

[6] http://en.wikipedia.org/wiki/Java_Native_Interface (July 26, 2011)
[7] http://www.ros.org/wiki/rosjava (July 26, 2011)
[8] http://www.robocupathome.org (March 16, 2015)

A basic home environment, supplied as a general scenario, provides at least a living room and a kitchen.

The European Robotics Research Network (EURON) Benchmarking Initiative[9] states that defining a benchmark is difficult and proposes that success should be related to quality, i.e. the robot really must serve its purpose and "benchmarks should focus on particular sub-domains, such as visual servoing, grasping, motion, planning etc."

Defining benchmarks in industry environments seems to work differently, since many resources for benchmark development can be provided. Furthermore this initiative gives an overview of current efforts in comparative robotics research and provides some initial results[10]. A classification of benchmarks as either analytic or functional is proposed. In both categories a benchmark can have either component or system level focus. [Bal00] gives an introduction to the benchmarking of mobile robots with respect to robot capability-specific tasks. [ILSKa05] focuses on reproducible test specifications and evaluation criteria and also on benchmarking. It formalizes tests and presents them using domestic grasping and placement tasks.

In [vdZI11], a benchmarking approach in the context of RoboCup@Home provides a cognitive benchmark for mobile robots where the order of task execution is left open in order to allow the robot flexibility when addressing new situations. In addition to cognition, it also provides benchmarks for robot capabilities, e.g. person recognition, object recognition, object manipulation, speech recognition etc. Due to yearly participation of robotics groups in this event, results are already available for more than seven years beginning in 2008.

[HIvdZ13] discusses the requirements for the redefinition of benchmarking principles for the testing of domestic service robots, specifically when it comes to dynamic and uncertain environments that include humans. The work also explains the evolution of the RoboCup@Home statistical procedure to track and steer progress since 2006. An important recent milestone is, according to the authors, the benchmarking of robot cognition, i.e. the ability to understand and reason about the world. These tests involve for example an undefined order of deliberative actions and complex requirements for speech recognition. The work concludes that beside essential capabilities (navigation, perception), there has been significant progress in all fields in recent years. Older reports of this contest, such as [WvdZIS09, WvdZIS10], focused strictly on robot capabilities, such as person or object recognition, manipulation or mapping. [BGMH+12] presents *Johnny*, an autonomous mobile service robot with an ontology and plan-based control architecture. It has successfully taken part in the RoboCup@Home challenge. Individual robot capabilities are nevertheless evaluated.

The book [RM09] on performance evaluation and benchmarking of intelligent systems tries to present a comprehensive overview as the first book of its kind on the topic.

[9] http://www.euron.org/activities/benchmarks/index (March 16, 2015)
[10] http://www.cas.kth.se/euron/euron-deliverables/ka1-10-benchmarking.pdf (March 16, 2015)

Chapter three focuses on assistive robots in the medical care area, specifically on end-user evaluations and on performance measures using a Fitness to Ideal Model (FIM) approach. It concludes that performance measures should be dedicated to the relevant domain and task at hand. Chapter six draws the connection between simulation and real robots. According to the authors, simulation does indeed accelerate robot development but simulation is also "doomed to succeed" as most simulation fails to simulate the unexpected. The authors present a methodology and techniques for simulation development with one of today's leading simulation engines.

[Wis09] is related to RoboCup@Home and concentrates on benchmarking in scientific competitions for service robotics. It addresses the problem of defining standard benchmarks for quantitative performance evaluation of mobile service robots. It is intended for non-standardized environments in the real world (including human-robot interaction). It features co-evolutionary development processes between test procedures and robot capabilities and evaluates robot capability (sub-) levels to create a final score.

The Defense Advanced Research Projects Agency (DARPA) Grand Challenge has high public and media recognition and is very application-oriented.

Several interesting evaluation approaches for intelligent systems exist. However they are typically focused on a dedicated use case, e.g. RoboCup@Home, medical care etc. Therefore use-case dependent benchmark criteria for measuring the benefit of an imagination based robot control system will be presented in Chapter 6.

2.5 Summary

There has been much recent work on the development of plan-based systems and on extending current approaches with so called semantic attachments in order to reason about geometric, spatial, temporal and causal aspects. First approaches show how simulation can help to train such systems for better parametrizations. Given the available planning, simulation, perception and robot control possibilities presented in this chapter, an integration of reasonably chosen instances of each field promises to be an efficient foundation for a physics-based prediction approach. To evaluate intelligent and learning systems, new approaches and metrics have to be developed, as the methods presented here are either of limited scope or cannot benchmark the benefit of autonomous competence enhancement. The next chapter, Chapter 3, focuses on the experimental platform used for this work.

Experimental Platform

3

This chapter introduces the physical robot platform used for experiments and evaluation. The platform has a variety of sensors and two versatile arms, each with a two-finger sensory gripper. Extra sensors have been mounted to augment the standard sensors with additional modalities. An open-source software framework is introduced. The framework helps with robot control and sensor data acquisition and with the development of software packages. Relevant hardware specifications are introduced and a description of the experimental physical domain, including mapping and navigation, concludes this chapter.

3.1 The PR2 Robot

In early 2012 when the research for this thesis started, the PR2 was probably the mobile robot most commonly used by the robotics research community. It had recently been added to the range of robots available at the Technical Aspects of Multimodal Systems (TAMS) laboratory. The PR2 robot was introduced by Willow Garage in 2010 as a successor to the Personal Robot 1 (PR1) prototype developed at Stanford University in the context of the personal robotics program[1] in 2007 [WBVdLS08]. Willow Garage was founded in 2006 by Scott Hassan, a prolific software engineer and an early Google employee, as a research lab dedicated to robotics. At that time a team of researchers was put together with the goal of impact first, return on capital second. Though labelled an experimental platform, the PR2 is intended for practical robot applications, e.g. serving in a restaurant as a waiter or, in personal life, doing laundry or supporting humans in health or elderly care. With the ROS software framework, the PR2 allows new users to rapidly prototype an application, e.g. within a week. A main driver behind the PR2 is a world-wide community working on the same hardware platform. This helps to avoid reinventing the wheel and allows a focus on new capabilities that can be shared among PR2 users. Last but not least, it is easy to reproduce results, which enables true scientific validation. At the time of writing there are 34 PR2 users world-wide[2].

[1] http://personalrobotics.stanford.edu (March 16, 2015)
[2] http://www.willowgarage.com/pages/pr2/pr2-community (March 16, 2015)

3.2 Actuators

In order to manipulate its environment, a robot needs to move in a human environment and to handle objects. Holonomic robots have a number of controllable degrees of freedom equal to the total degrees of freedom in the movement of the base. The PR2, with four active casters each of two wheels (see Fig. 3.3), is considered to be a holonomic omnidirectional robot [SK08], although this is theoretically not true as the wheels might have to be rotated before the base can move in any particular direction. Examples of "real" omni wheels are, for instance, the Mecanum wheel[3] and ball transfer units[4]. The robot operates in two dimensions and thus the (base) system state is defined as $x = [x\ y\ \theta]^T$, where the robot position is represented as $x, y \in \mathbb{R}$ and its orientation as $\theta \in \mathbb{R}$. The control input is $u = [u\ v\ w]^T$ with $u \in \mathbb{R}$, $v \in \mathbb{R}$ and $w \in \mathbb{R}$ being the linear and angular velocities of the robot. To derive practically feasible trajectories, the following conditions should be satisfied:

$$\begin{aligned} 0 < \|u\| \leq u_{max},\ \|a_u\| \leq a_{u\,max} \\ 0 < \|v\| \leq v_{max},\ \|a_v\| \leq a_{v\,max} \\ 0 \leq \|w\| \leq w_{max},\ \|a_w\| \leq a_{w\,max} \end{aligned} \tag{3.1}$$

Where a_u, a_v, $a_w \in \mathbb{R}$ denote the linear and angular accelerations, respectively. u_{max}, $a_{u\,max}$, v_{max}, $a_{v\,max}$, w_{max}, $a_{w\,max} \in \mathbb{R}$ represent the corresponding maximum values of velocity and acceleration u, a_u, v, a_v, w, a_w.

The two, 7-DOF, arms allow versatile manipulation of objects in a large workspace, as shown in Fig. 3.4. Fig. 3.1 shows that each arm consists of four passively counterbalanced, back-drivable, joints; the wrist includes three joints and the gripper adds another DOF. In combination, the upper arm and forearm give a practical arm length of about $72cm$. The wrist offers $4Nm$ of torque and the grip force is $80N$. The arm payload is around $1.8kg$. A remarkable feature is that both the forearm and wrist roll-joints can operate continuously. Back-drivability and the counterbalance mechanism as well as the soft arm cover (not including the gripper) enable the robot to operate safely in human environments. The counterbalance mechanism ensures that the arms do not drop in case of a power cut or emergency. Back-drivability is very important for the avoidance of injury. It gives an aspect of resilience (as opposed to rigidity) to the robot arm in that it allows a human user to move the arm by pushing on any part.

The fingertips are covered with tactile sensors. Each finger has 22 tactile cells that support discrete force readings. This enables accurate manipulation (corrections) as it is possible to detect when an object has been grasped stably and safely. The sensor

[3] http://en.wikipedia.org/wiki/Mecanum_wheel (March 16, 2015)
[4] http://en.wikipedia.org/wiki/Ball_transfer_unit (March 16, 2015)

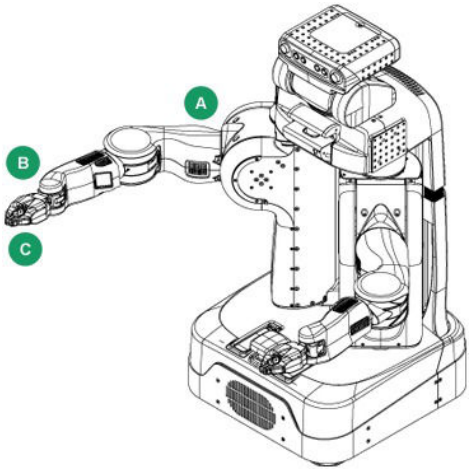


Fig. 3.1: PR2 arm joint specifications: (A) Arm: 4-DOF (passive counterbalance, backdrivable), (B) Wrist: 3-DOF, (C) Gripper: 1-DOF. Upper arm: 400mm, forearm: 321mm. Payload: 1.8kg, wrist torque: 4Nm, grip force: 80N. Continuous forearm and wrist roll joints. [Willow Garage]

layout is shown in Fig. 3.2. The rubber finger tip surfaces allow objects to be held securely without slipping.

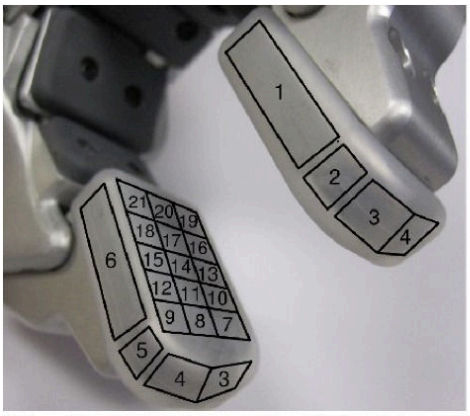


Fig. 3.2: PR2 gripper sensors: Arrangement of the 22 pressure sensors on each finger pad.

To maximize manipulation capability while maintaining a small driving posture (to avoid obstacles while driving), the upper torso of the PR2 can be extended upwards by about 30cm. Raising the torso increases the workspace accessible by the two arms when manipulating objects on a table. This is discussed further in Chapter 6 (Section 6.2.1). The full extension of the torso is shown in Fig. 3.6 and the specifications are shown in Fig. 3.3.

The PR2 provides two *i7* Xeon based server computers each with eight Central Processing Unit (CPU) cores. This, combined with 24 Gigabyte (GB) of memory, makes

on-board sensor data processing possible. In fact the base system (Hardware (HW) drivers, manipulation control, navigation and collision avoidance) runs completely on board. Four hard disk drive (HDD) storage devices allow the logging of various sensor data for testing, debugging or later processing.

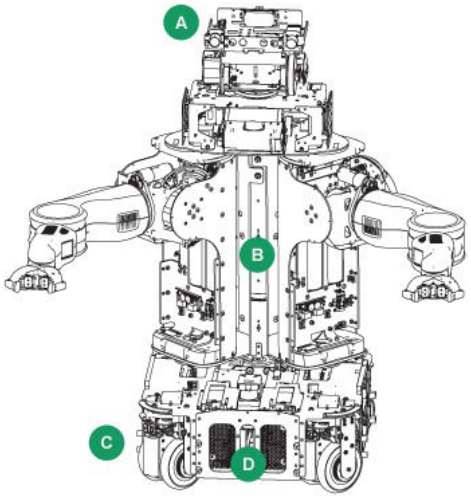
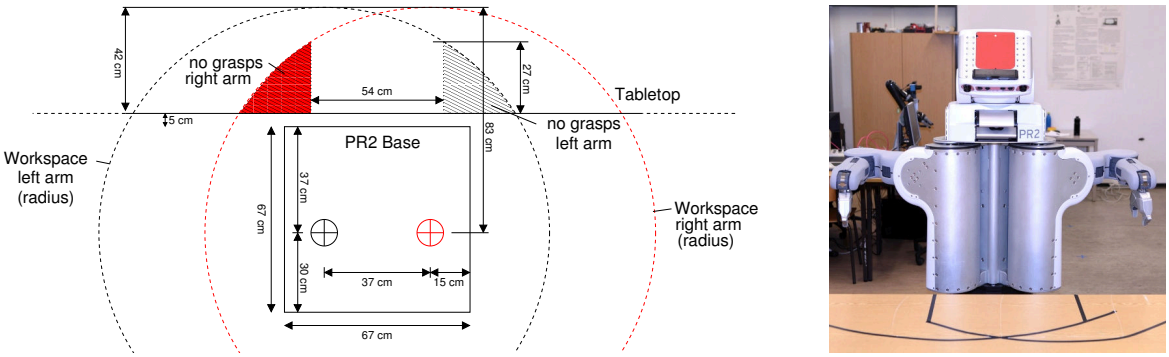


Fig. 3.3: PR2 base and computing: (A) Pan-tilt head, (B) Telescopic spine (from 1330 to 1645 mm), (C) Omni directional base: 4 steered and driven casters, max. 1 m/s, (D) On-board servers: two Quad-Core i7 Xeon (PR2 on-board computer 1 (C1) and PR2 on-board computer 2 (C2), 8 cores each), 24 GB Random Access Memory (RAM), two 1.5 Terabyte (TB) removable hard drive, two 500 GB internal hard drive. [Willow Garage]

For practical use the actual work space, e.g. on a table, is important. Fig. 3.4 shows that the measured work space covered by the two arms is a rectangular area of about $54cm$ by $27cm$ in front of the robot when the robot base is $5cm$ in front of the table. When considering the PR2's arm length in addition to its shoulder, the resulting radius without the gripper is about $83cm$.

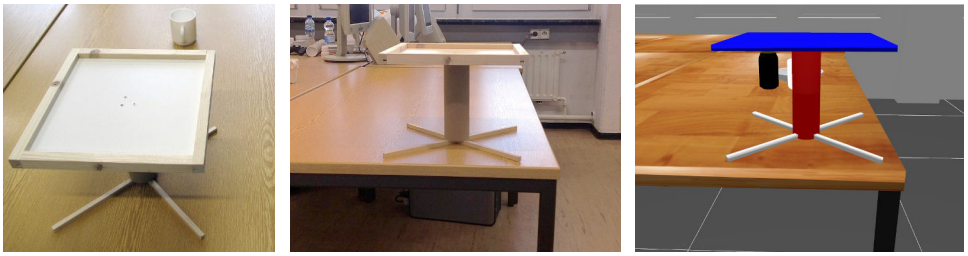
3.3 Tray

Service robotic tasks might involve carrying multiple objects. A tray allows the PR2 to carry more than the two objects it can carry in its hands. An initial design was developed that was grasped by the robot. This design is shown in Fig. 3.5. The tray is meant to be grasped while objects are transported on top. With this design, the arm-joint tolerances and inadequate contact between the PR2 gripper and the tray handle produce wobble when the robot is moving. This instability and difficulties in grasping the tray made the design unsuitable, despite the benefit of being able to leave the tray on a table (which allows the robot to move without the tray). The design was therefore rejected in favor of the robot mounted design shown in Fig. 3.6.



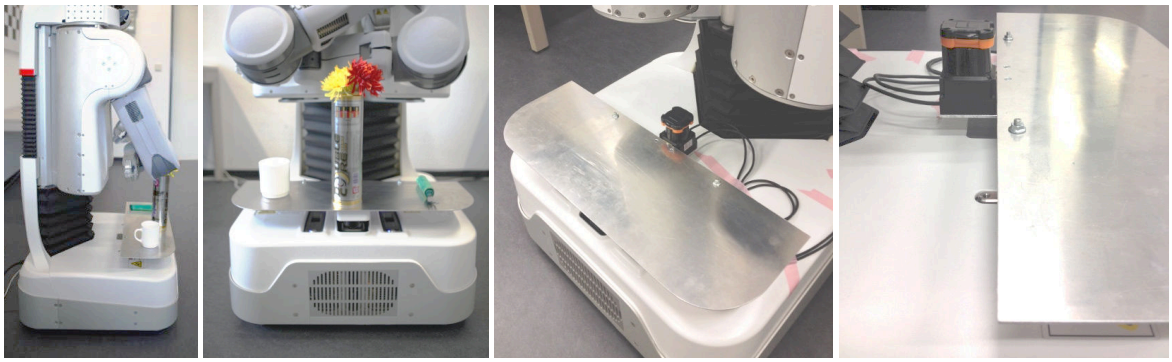
(a) (b)

Fig. 3.4: PR2 arms work space: Standing in front of and close to a table, the robot has a rectangular work space reachable by both arms of 54*cm* by 27*cm* (a). This work space was measured with a table height of 74*cm* (b).



(a) (b) (c)

Fig. 3.5: First PR2 external tray design (a), (b). A physically and visually simulated model (c).



(c) (d)

Fig. 3.6: Final PR2 mounted tray design with laser scanner to detect objects on the tray (c), (d)

Even though the tray is not removable it should not limit the robots work space or normal operation. A flat design at the robot's base was a reasonable solution. This design does not extend the robot's footprint and thus does not increase the likelihood of collision with its environment.

3.4 Sensors

The PR2 robot comes with a variety of sensors on board. There are two front-pointing laser scanners, one mounted in the base and one near the head. These laser scanners allow detection of objects in the robot's environment at distances of up to $30m$ and with a viewing angle of 270°. They are used for mapping and collision avoidance. The head includes two stereo camera systems for 3D and 2D vision, with a red light projector to support stereo camera accuracy. In addition, a high definition (HD) camera is available. All default sensors are shown in Fig. 3.7.

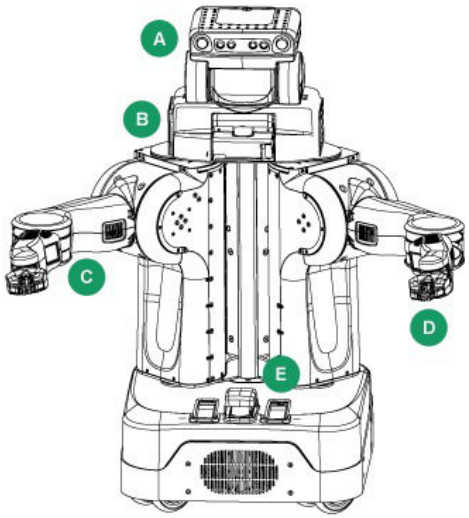


Fig. 3.7: PR2 default sensors: (A) 5 Mega-pixel (MP) Ethernet color camera, wide-angle color stereo camera, narrow-angle monochrome stereo camera, Light-emitting diode (LED) texture projector triggered with narrow-angle camera, (B) Tilting Hokuyo UTM-30LX laser scanner, 3D Inertial Measurement Units (IMU), (C) two forearm cameras, (D) Gripper: 3-axis accelerometer, fingertip pressure sensor arrays, calibration LED, (E) base: Hokuyo UTM-30LX laser scanner. [Willow Garage]

Extra sensors were added to support additional modalities. A Kinect-like sensor, the *Asus Xtion Pro Live*, is mounted on top of the robot head as shown in Fig. 3.8. This sensor can replace the stereo camera systems as it has lower latency and less overall noise. A thermal camera was also attached to allow the detection of object temperatures. Both sensor mounts were modified to attach to the PR2's head frame.

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

Fig. 3.8: PR2 mounted additional sensors: Asus Xtion Pro Live (a,d) and Flir A615 thermal camera (b,c)

Internally there is no difference between the Microsoft Kinect and ASUS Xtion Pro Live sensors, as both use the same *PrimeSense* sensor technology. The basic specifications are: depth range between $0.8m$ and $3.5m$, a field of view of 58° horizontal, 45° vertical, 70° diagonal and a resolution of 640x480 at 30 frames per second (FPS) red, green and blue color space (RGB) and depth [ASUS]. Audio sensing is not used in this work. The advantages of mounted sensors are lower power consumption, a smaller casing and a $5V$ supply over USB (no external power supply is needed). The lack of a tilt motor is compensated for by the tilting joint of the PR2's head.

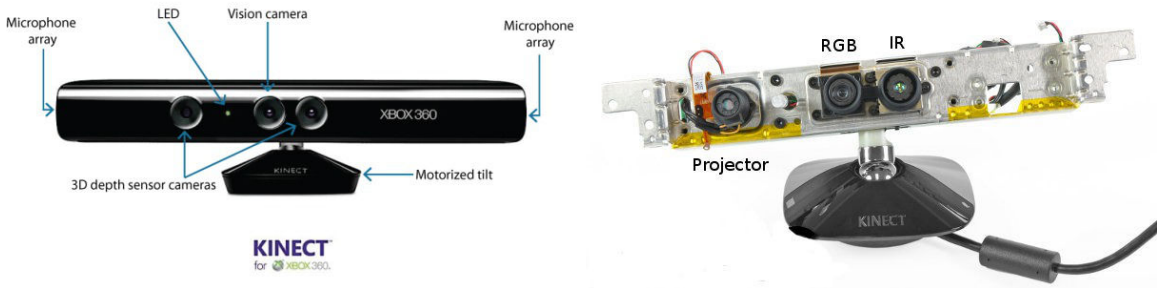


Fig. 3.9: The Microsoft Kinect sensors and actuators

The PR2's on-board visual and inertial sensors function as the robot's eyes and inner ears, allowing the robot to keep track of its speed and orientation. To obtain exact measurements, calibration of the relative positions of the robot's cameras and of the IMU is necessary. The consequence of inaccurate calibration would be 'disagreement' about the robot's motion causing navigation problems. Therefore, regular calibration of the robot is required, which is typically time-consuming and labor-intensive. The PR2 has a semi-automatic procedure for calibrating its sensors. The calibration results shown in Fig. 3.10 are of a 'good' calibration used in this work, including the additional Kinect sensor. This calibration was performed with the PR2 calibration package[5]. At the time of writing an experimental calibration procedure was available but that was not tested in this work.

[5] http://wiki.ros.org/pr2_calibration (March 16, 2015)

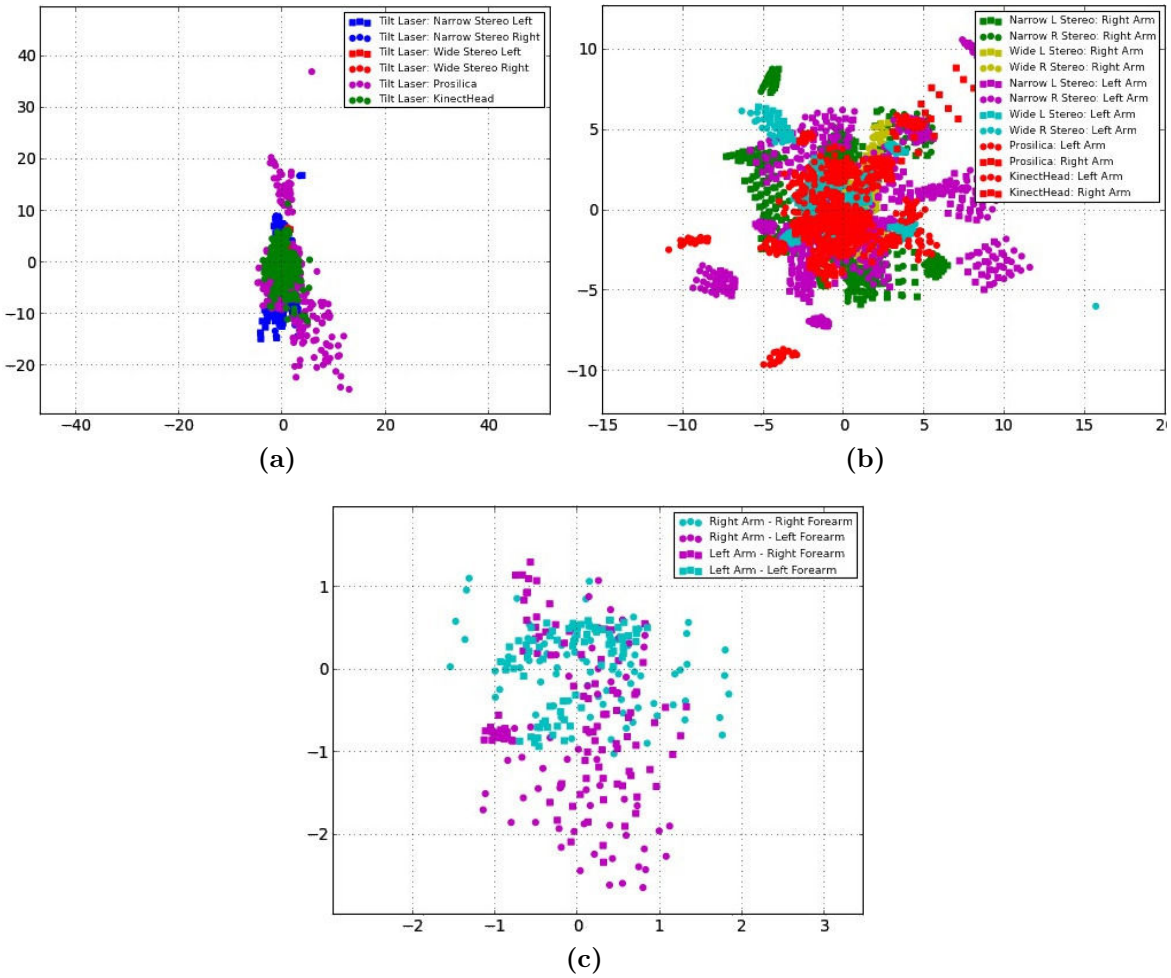


Fig. 3.10: PR2 calibration results: error values are in pixels. (a) PR2 head sensors vs. tilt laser. (b) PR2 head sensors vs. right and left arm. (c) right and left vs. right and left forearm camera

For calibration, transformations between the PR2's local coordinate frames are used. The transformations for all standard PR2 links and sensors are shown in Fig. 3.11.

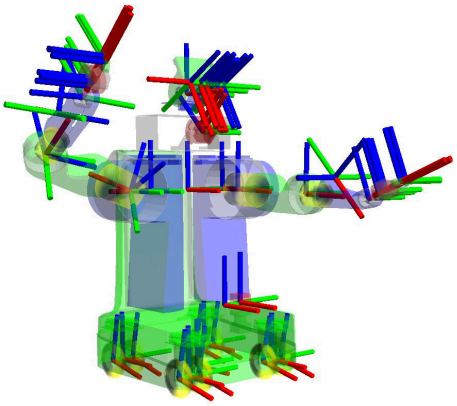


Fig. 3.11: PR2 coordinate frames: the configuration used has 99 frames, depending upon the attached sensors.

3.5 Power and Control

A typical ROS-based system consists of many Operating System (OS) processes. Processes and inter-process communication rely on OS functionality that is not in principle real-time capable on a non real-time OS, such as Windows, Linux, Mac OS X[6]. To communicate with HW components, a low level software layer operates in a $1kHz$ loop. Dedicated motor controller boards operate in a $100kHz$ loop. The latter communicate directly with the robot's motors and encoders. The basic control flow, illustrated in Fig. 3.12, shows the abstraction of control elements. One important aspect is the transition between non-real-time and real-time levels at the Controller Manager.

Fig. 3.13 shows the network setup including the base station computer and the PR2 robot. The robot's internal setup shows its two computers, a local area wireless technology (WiFi) router and an additional wireless access point (WAP). A sophisticated tunneling mechanism allows transparent communication between computers on the building network and the robot, whether the PR2 is connected wirelessly or via Ethernet cable. More details can be found in the PR2 manual[7].

The PR2's power system consists of a $1.3kWh$ lithium ion battery pack allowing an approximate run time of 2 hours depending upon the robot's task. When operation requires a longer run time or the battery has decreased in capacity over its lifetime, it is possible to charge and run the robot at the same time. For safety, an on-board

[6] Although there might be real-time capable OS distributions for special purposes, such as embedded systems.

[7] https://pr2s.clearpathrobotics.com/wiki/PR2%20Manual (March 16, 2015)

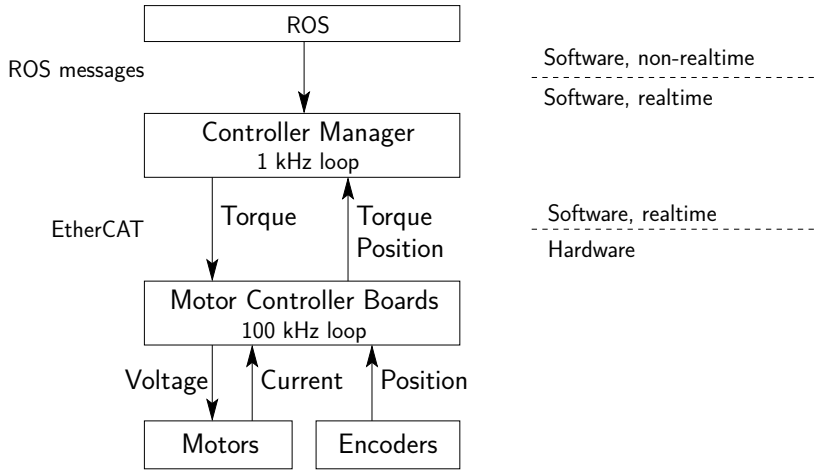


Fig. 3.12: PR2 low-level motion control [Willow Garage]

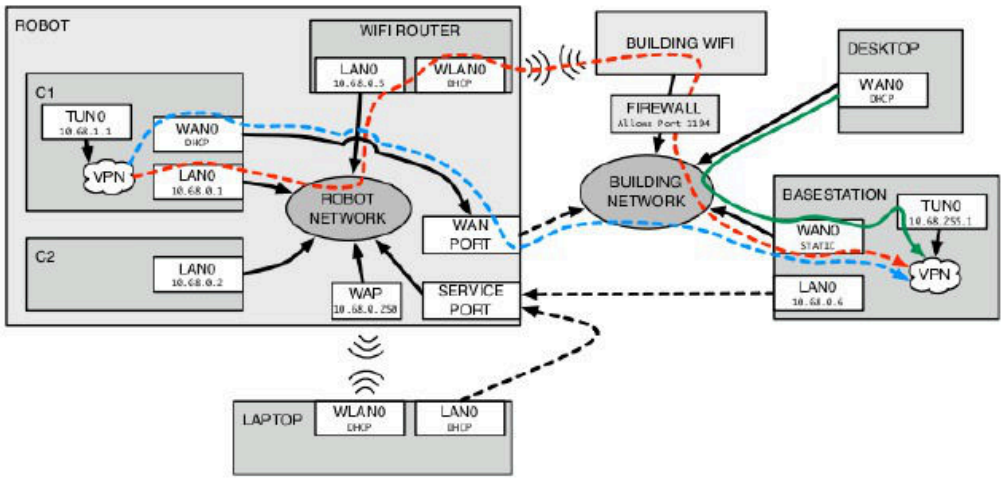


Fig. 3.13: PR2 network: Robot: 32 Gigabit back plane switch, multi gigabit connection to each server, separate camera gigabit network. 1kHz control to all motors. Dual-radio WiFi, dedicated service access point, Bluetooth access point. [Willow Garage]

and a wireless 'run-stop' (cutout switches) allow instant disabling of all motor drivers while still leaving all the sensors, servers and motor board diagnostics running [Willow Garage].

3.6 Navigation and Mapping

Navigation is an important capability for a mobile service robot. In this work the standard ROS PR2 packages are used. These contain navigation related methods [MEBF+10], which are: an implementation of the A* algorithm [Kon00] for global path planning; and a Dynamic Window Approach (DWA) implementation, see section 4.4, for local path planning. Also used are implementations of the Adaptive Monte Carlo Localization (AMCL) [FBDT99] probabilistic particle filter and of the extended Kalman filter [Kal60] to fuse IMU with wheel odometry data. In order to detect obstacles on the ground the Random Sample Consensus (RANSAC) algorithm [FB81] is used to extract a (horizontal) plane (e.g. the floor) from the laser scan; outliers are classified as obstacles.

As a basis for global 2D navigation, an occupancy grid map of the 3rd level TAMS floor of the F building on the computer science campus[8] was created with the occupancy grid Simultaneous Localization And Mapping (SLAM) technique [TBF05]. The commands to launch the SLAM system on the PR2 can be found in Appendix A.1. A standard ROS package[9] combines the SLAM algorithm with the standard PR2 navigation algorithm[10] [MEBF+10]. The resulting TAMS floor map is shown in Fig. 3.14. In the map are the main floor and the laboratory, which houses the restaurant environment. In addition to the map-origin coordinate frame, other origins are shown. This map can be used as a blueprint for building an accurate 3D environment, as shown in Chapter 5.

3.7 Summary

Willow Garage closed in 2013. Many successful ideas have been spun out as commercial start-ups, such as *Savioke* (autonomous service robots) and *Unbounded Robotics* (low cost mobile manipulation robots), or as independent non-profit organizations, such as the Open Source Robotics Foundation, Inc. (OSRF) and more.

This chapter gave an overview of the PR2 platform. It showed that the platform is well-suited for the tasks envisaged, mainly because of its versatility and multimodal sensors. The huge open-source community working with the platform and the

[8] https://www.inf.uni-hamburg.de/en/service/location.html (March 16, 2015)
[9] http://wiki.ros.org/pr2_2dnav_slam (March 16, 2015)
[10] http://wiki.ros.org/navigation (March 16, 2015)

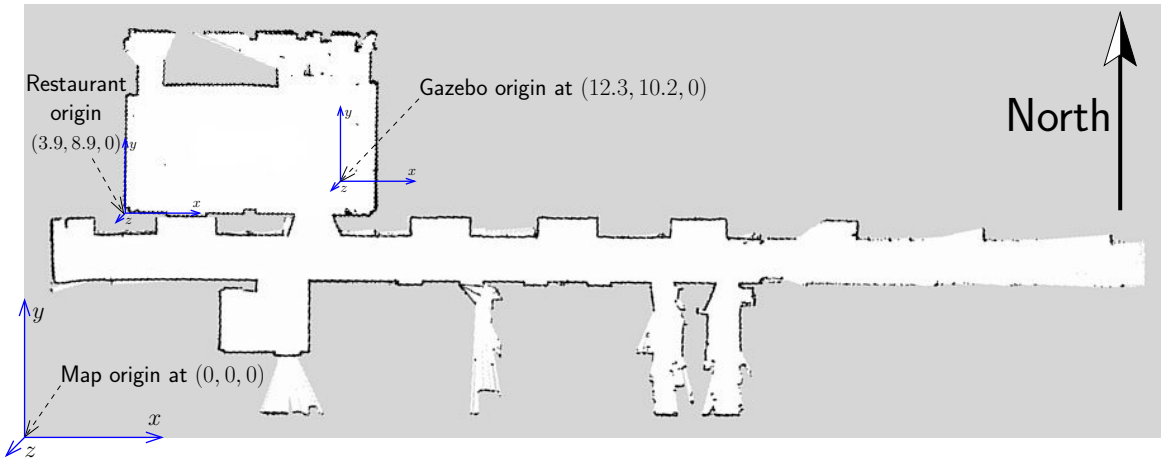


Fig. 3.14: 2D occupancy grid map of the TAMS floor including the Lab is the navigational basis for the RACE domain at a resolution of $0.05cm$ per pixel. 'North' defines the map orientation in relation to the world coordinate frame.

availability of software frameworks and tools all support its application in this work. In addition, the necessary additional sensors and tools have been described and important specifications have been listed. Sensor calibration and map building results have been discussed. The next chapter, Chapter 4, focuses on the frameworks and tools used in this work.

Frameworks and Tools

4

This chapter introduces the software frameworks employed and developed in the course of this work. The RACE project is introduced first, as it not only provides a development framework but it also serves as the foundation of the scenarios and ideas of this work. Related parts of ROS, such as the underlying development framework, will be discussed. An important tool is the Gazebo simulator and its physics-based engine, ODE. These will be discussed, along with two frameworks developed to create toppling experiences, to generate unsupervised experiences in changing environments and to create and record training data.

4.1 The RACE Project

In previous work, an ontology-based robot control architecture was developed to allow robots to learn from experience [RNZ+13]. To focus the research on high-level tasks, such as learning from experience employed in the RACE project[1], it is necessary to be able to take the robustness and functionality of lower level components for granted. The platform provides basic capabilities, such as navigation, manipulation and grasping.

The three year EU funded FP7 project RACE started in December, 2011 and ended in November, 2014. Most of the work of this thesis was developed in the context of this project. The overall aim of the project was to enhance the behavior of an autonomous mobile service robot by having the robot learn from conceptualized experiences of previous performances, based on initial models of the domain and on its own actions. The contributions of the project to the state of the art can be summarized as producing a robot:

- that is capable of storing experiences in a multi-level way using appropriate structures,
- that is able to learn and generalize,
- that demonstrates enhanced competence by applying learned knowledge.

[1] http://www.project-race.eu (March 16, 2015)

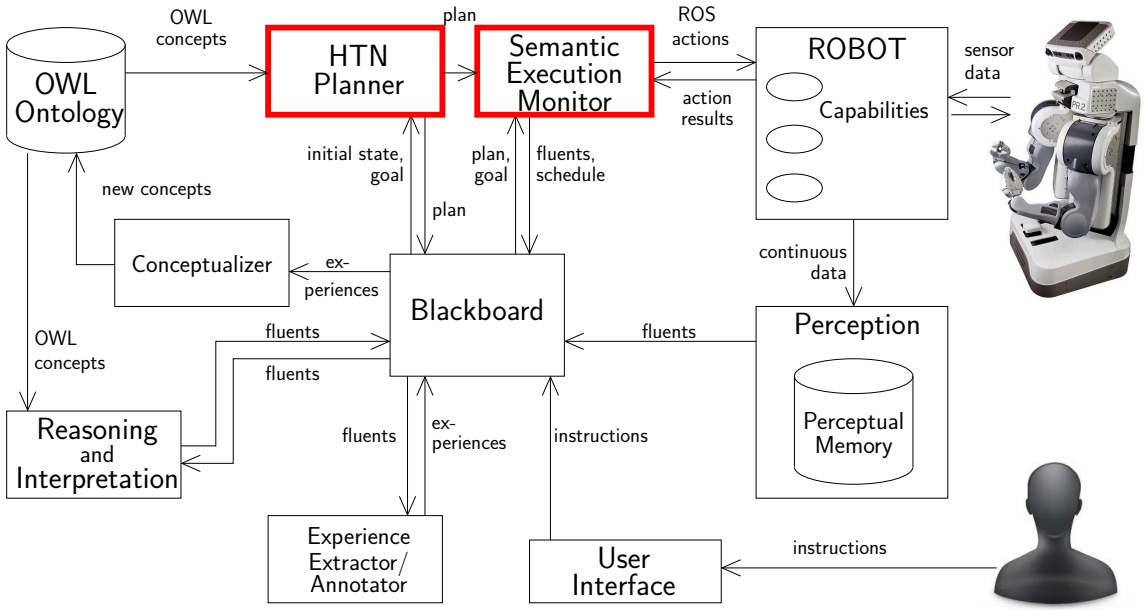


Fig. 4.1: The RACE software architecture. The modules closely involved in functional imagination are highlighted.

A high-level system architecture is shown in Fig. 4.1. A more detailed description of the system, its parts and the evaluation results is given in [HZZ+14, RNZ+13]. Work that focuses on evaluation is described in [ZRP+13, ZRP+12]. The resulting system is plan-based. It adapts its behavior using hybrid reasoning causally, temporally and spatially. This leads to robust real-world task execution. Nevertheless the most important achievement is the enhancement of robot competence by learning from experience. This enhancement is measured in terms of performance quality and the description length of the robot instructions.

4.2 ROS and the PR2

Robots and robot control software become ever more complex. There is also a tendency to use ever more sensors and consequently the amount of data to be processed rises. As the complexity of tasks increases and as tasks are divided into many parts, coherence between capabilities grows rapidly. For example the PR2[2] robot produces data at about $50MB/s$, not including the RGB-D sensor output. Newer sensors with higher resolution, such as the second generation Microsoft Kinect, already produce 3D data at a rate of about $60MB/s$.

ROS provides many features for the PR2 as part of the standard system, although most of the features are platform independent and can be used on other robots as

[2] http://www.willowgarage.com/pages/pr2/overview (March 16, 2015)

well. Furthermore, tools providing additional functionality, e.g. image processing, often have a ROS API and cooperate well with ROS. Examples are the Open source Computer Vision library (OpenCV)[3] and the Point Cloud Library (PCL)[4]. Both software frameworks provide advanced perception filtering capabilities, such as 2D and 3D image processing. The standard ROS tools include applications for navigation, manipulation, visualization, logging and playback, simulation and hardware diagnostics. A running ROS instance typically maintains many 'nodes' - processes in a graph-like topology communicating with each other using streaming topics, Remote Procedure Call (RPC) services and a parameter server. The node hierarchy is flat, consisting of a master node and all other nodes. The master node is responsible for all communication requests between nodes. It also acts as a network time server and both this and limitations of the parameter server lead to difficulties in distributed systems. These are discussed in Section 5.3, together with possible solutions.

In the development phase, Willow Garage (originators of the PR2) used software components from the Player Project[5] to develop the application programming stack for the PR2 robot. Its three components, Player, Stage and Gazebo, have all been adapted for use within ROS. Thus important parts, such as drivers, algorithms and simulation environments could be reused. The Player Project [GVH03] grew from the need for alternatives to existing closed source software and was developed in 2000 initially by Brian Gerkey, Richard Vaughan and Kasper Stoy. It was hosted at the University of Southern California (USC) Robotics Lab website before it migrated to SourceForge.

4.3 Gazebo Simulator

The Gazebo[6] simulation is used in this work. The version used (1.0), which was released early in the project (2012) and retained, brought major updates compared to the previous version (0.1), such as the completely re-implemented Graphical User Interface (GUI) with "Cute" cross-platform programming library (QT4) and performance improvements, as well as an adapted Extensible Markup Language (XML) world file syntax.

The simulator features[7] related to this work are:

- the ability to simulate indoor environments with multiple robots in real-time
  - multiple rooms (at least three), multiple objects, one PR2

[3] http://wiki.ros.org/vision_opencv (March 18, 2015)
[4] http://www.pointclouds.org (March 18, 2015)
[5] http://playerstage.sourceforge.net (March 18, 2015)
[6] http://gazebosim.org (March 18, 2015)
[7] http://wiki.ros.org/gazebo/Version_1.0_Design_Specification (March 18, 2015)

  - a physics engine that allows stable manipulation of objects during run time
- the graphical editing of worlds (not used; XML was edited directly)
  - ability to import common, self-created meshes in Collada or Surface Tessellation Language (STL) format
  - creation of robot models: ability to set joints and physical properties
  - ability to apply forces and torques to models
  - ability to position objects via the GUI
  - a camera, for 2D rendering
  - visualization of object collision models
- a ROS API

The current implementation uses ODE as a physics engine, the Object-Oriented Graphics Rendering Engine (OGRE) for rendering and QT4 as the GUI. Sensor and controller access is provided via a shared memory interface, as indicated in Fig. 4.2. Gazebo simulates a 'World' that consists of an arbitrary number of models; each model can consist of joints, sensors or body elements, such as a box.

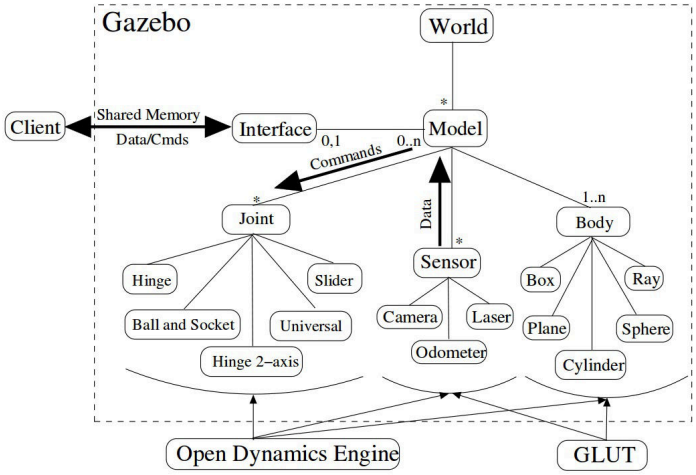


Fig. 4.2: General structure of Gazebo components [KH04]. Here the OpenGL Utility Toolkit (GLUT) is used as the Open Graphics Library (OpenGL) API.

Two threads run in Gazebo. A distinction is made between the GUI and rendering in one thread and the physical engine in a separate thread. This approach allows the physics engine to cycle at a higher rate than the GUI, which is necessary for high accuracy simulation.

The monolithic architecture prevents advantage being taken of multi-core and distributed systems and reduces the flexibility of Gazebo as a physics simulation. This

limitation on systems that wish to simulate multiple instances or that have strong real-time requirements influences the resulting architecture and design.

Figure 4.3 shows the architecture current when Gazebo 1.0 was released. In comparison, the ROS bundled version distinguishes between only two executables: the Gazebo server (including physics, rendering, sensors) and the client GUI. Note however that in the final 1.0 version *wxWidgets* was replaced by QT4.

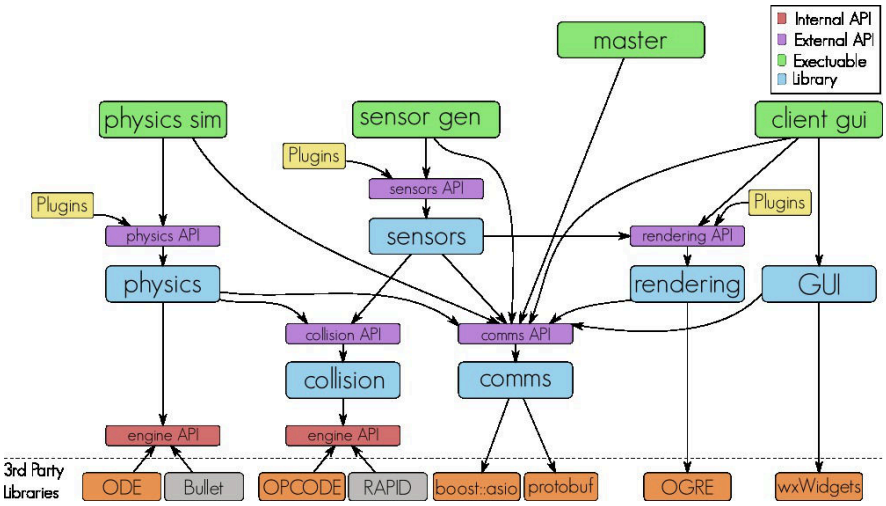


Fig. 4.3: Gazebo dependency graph [ROS.org].

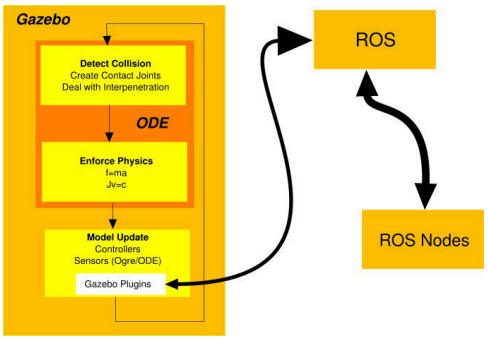


Fig. 4.4: Gazebo plug-in architecture

Gazebo provides an external API to alter physical world properties, such as setting the real-time factor. The API can also change individual items in the simulation, for example manipulating the pose of an object. The physics, sensor and rendering libraries support plug-ins that allow dedicated behavior or appearance control. For example it is possible to attach an object with its own dedicated model of inertia or to provide more sophisticated rendering effects by using Graphics Processing Unit (GPU) functionality. The Gazebo plug-in architecture is shown in Fig. 4.4. It should be noted that Gazebo (the ODE physics engine) is a rigid body simulator and does

not simulate soft objects - clothes or liquids, such as water. This limitation can be partially offset by using particles (e.g. small spheres) to approximate the behavior of liquids. This approach was applied, for example, in [KDB11].

4.3.1 Physical Simulation

Euler's method [BW97] is often used as an integrator, such as in ODE:

$$x(t_0 + h) = x_0 + h\dot{x}(t_0) \tag{4.1}$$

ODE supports static and kinetic friction as well as restitution, whereas Bullet does not provide kinetic friction. The Bullet physics engine instead supports partial graphics processing on the GPU for physics calculation. ODE is a constraint-based physics engine with fixed time-stepping. The evaluation compares accurate ODE integrator calculations for the costs of performance to those for Bullet. Tests include the displacement of a non-moving object compared to classical physics, which is:

$$r = \frac{1}{2}at^2 \tag{4.2}$$

where r is the body's displacement, a is the body's acceleration and t is the time.

Restitution is given in classical physics by:

$$C_R = \sqrt{\frac{h}{H}} \tag{4.3}$$

where C_R is the coefficient of restitution, h is the bounce height and H is the drop height. Friction is typically calculated according to the Coulomb friction model [KEP05].

The accuracy and performance of the physics engine when calculating friction, collisions and constraints are important for predicting robot action results in a typical domestic domain. ODE and Bullet differ slightly in these disciplines, but both provide the required capabilities. Nevertheless, no currently available engine offers superior performance in all situations. Thus the prediction model has to take variations and minor deviations from true behavior into account. Not all tested engines support noise models.

[WHR10] evaluates the Bullet engine as an ingredient of physical reasoning. It investigates the prediction accuracy of a physics engine when simulating a robot arm pushing flat objects across a table. The authors recommend reasoning about goals and actions using physics simulations. Successful reasoning is considered to imply the need for a

strong understanding of dynamics and the effects of actions. Physics engines can complement classical rule-based reasoning systems by acting as a source of information or as an 'oracle'. This can help answer questions concerning the stability of manipulated objects, such as whether they might tip over or start to roll, or how to shift several objects without changing their configuration. The authors propose that simulation accuracy is a problem of selecting parameters for an optimal match to reality. The parameters most important for optimization by a cost function are the object's shape factor and friction between the object and the ground and fingers.

$$E(P, P') = \frac{1}{n} \sum_{j=1}^{n} \left\| c_j - c_j' \right\| \tag{4.4}$$

Here the error E of the real and simulated poses, P and P' respectively, is the sum of the deviations of all object corner points c_j in contact with the ground. The authors propose using a combination of the simplex algorithm [NM65] and differential evolution [SP97] to search for optimal parameter values.

The authors conclude that accurate prediction of real world interactions is possible with state-of-the-art physics engines. However their focus is on flat objects that are pushed across tables. Furthermore the accuracy of the particular engine used has been optimized to reduce errors by 50 percent when compared to default values. They propose employing the simulation for learning processes and combining it with qualitative and quantitative simulation.

4.4 Robot Capabilities

The perception capabilities of the PR2 are based on the Object Recognition Kitchen (ORK) framework, which uses the ROS tabletop module[8]. The procedure is described in [ML09]. Firstly, a (table) plane is extracted from the point cloud by the RANSAC algorithm [FB81]. Remaining points are clustered and each cluster is processed using the Iterative Distance Fitter (IDF) technique for rotationally symmetric objects. This compares the cluster with Computer-Aided Design (CAD) models stored in a *CouchDB* database and results in a confidence value for each possible object type. For object recognition, the recognizable objects have to be rotationally symmetrical. In the case of recognized objects, grasping relies on learned gripper configurations (multiple configurations per object can be stored in the database) in relation to the object. In the case of new objects, grasping relies on bounding boxes calculated around the object's segmented point cloud. For a recognizable object, the bias is that the object has the same upright orientation stored in the database.

[8] http://wiki.ros.org/tabletop_object_detector (September 8, 2015)

There are several coherent frameworks, like ROS, that integrate many different capabilities. One such capability is navigation, which typically consists of localization, path planning and robot control. The combination of 2D navigation and 3D collision avoidance is the state-of-the-art method in mobile robotics. 2D navigation provides the necessary performance; 3D collision avoidance the required fidelity.

The DWA is used for local path planning [FBT97]. DWA is a local robot navigation approach in two dimensional space[9]. It considers current and projected velocity and acceleration settings for linear and angular directions and requires a global planner, which is not of interest here. Nevertheless a global plan and cost map are required to produce velocity commands to control the robot's base. The algorithm implementation in use supports robots with a footprint that can be represented as a convex polygon or a circle. One important property is its support for dynamic changes of parameters such as velocity and acceleration. The basic DWA algorithm is described in Algorithm 4.1.

Algorithm 4.1 : Basic DWA Algorithm

```
1 Δx, Δy, Δθ, obstacles, trajectories Output : collision free velocity v
2 while true do
3     sample the robot's control space (Δx, Δy, Δθ);
4     foreach v do
5         perform forward simulation for Δt of robot's current state;
6     evaluate each trajectory score regarding proximity to obstacles, the goal, the
        global path and v;
7     discard trajectories with collisions;
8     pick highest-scoring trajectory;
9     return v;
```

Other essential capabilities are manipulation and grasping. For these it is important that a solution for inverse kinematics exists, in addition to solutions for the calculation of possible trajectories and grasps. One difficulty is the calibration between the sensors, the robot and the gripper pose. High grasp accuracy and the coordination of grasping with the recognition process must be ensured. Generally the task can be finished successfully only if all components are sound. If one component fails, the goal cannot be achieved. Other components, such as the planner, object recognition and the simulator are indispensable and belong to the basic capabilities.

[9] http://wiki.ros.org/dwa_local_planner (March 19, 2015)

4.5 Summary

This chapter gave an overview of the software frameworks and tools used in this work. The RACE project - the foundation of this work - was introduced, and its high-level architecture was presented. ROS, the underlying open-source development framework, was described, and PR2-related components were discussed. The Gazebo simulator, which is the basis of the functional imagination approach in this work, was introduced, and robot capabilities necessary for the experimental scenarios were presented. In the next chapter, Chapter 5, a detailed look will be taken at the approach taken by this work.

Functional Imagination

5

This chapter describes the methods and algorithms of the proposed system. It also describes the robot capabilities developed for the experiments. Experimental methods and scenarios are developed and the scenario setup is discussed in detail. The models required to represent the problem domain are developed and a detailed architecture is crafted.

The chapter also gives an overview of the physical realization of the approach in terms of software and hardware. It describes in detail how frameworks and tools are employed, combined and optimized. The goal of this chapter is to develop and integrate a simulation system that provides a basis upon which robot action results can be predicted and exploited for plan adaptation in order to prevent sub-optimal activity (cf. Section 1.4).

5.1 Methodology

A pragmatic approach to the use of physics-based simulation for predicting robot action results is to simulate an action and then to evaluate the resulting simulation state. The three important parts of the process are:

1. intention
2. simulated execution
3. simulated result

In this work, the intention is the action the robot is supposed to execute, i.e. the plan (step) created by the symbolic planner; the simulated execution is the manipulation of the world state, i.e. execution in Gazebo; the simulated result is the world state in the simulation after the execution has finished.

5.1.1 Traditional Robot Control Paradigms

Since around the 1970's, the predominant robot control paradigm has been Sense-Plan-Act (SPA). The robot is supposed to sense its environment, create a plan of what to do next and execute that plan. This is a continuous cycle during the robot's lifetime. The robot operates in a hierarchical/deliberative fashion, heavy on planning and storing gathered sensor readings in a global world model. Fig. 5.1 shows this paradigm.

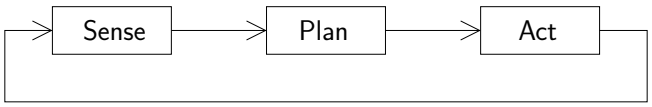


Fig. 5.1: Hierarchical Sense-Plan-Act robot control paradigm

At the lower level, such as for the reactive execution of manipulation or locomotion, the Act-Sense paradigm of reactive robot control is typically applied. In this, the robot has multiple, concurrent, act-sense couplings (Fig. 5.2). Each coupling creates an output, i.e. the robot's behavior, independent of other couplings. The robot eventually executes a combination of these behaviors. A prominent example of the reactive robot control paradigm is the *Subsumption* architecture [Bro86].

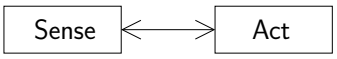


Fig. 5.2: Reactive-Sense-Act robot control paradigm

Today's robot systems typically apply a combination of these two paradigms (Fig. 5.3). This hybrid deliberative/reactive paradigm takes advantage of the hierarchical decomposition by the planner of one task into subtasks and the reactive accomplishment of each subtask. Exactly this approach is applied within the RACE framework with its plan-based nature and its reactive ROS robot capabilities.

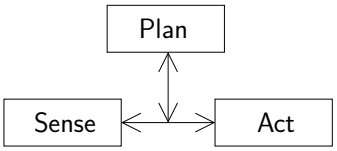


Fig. 5.3: Hybrid Sense-Plan-Act robot control paradigm

5.1.2 Hybrid Sense-Plan-Imagine-Act Control Paradigm

In hierarchical robot control, imagination would be a prerequisite for acting and a consequence of planning, as shown in Fig. 5.4. Consequently a hybrid Sense-Plan-Imagine-Act would involve imagination as shown in Fig. 5.5.

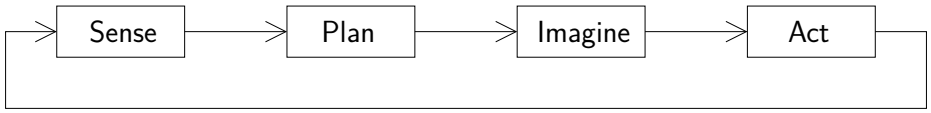


Fig. 5.4: Hierarchical Sense-Plan-Imagine-Act robot control paradigm

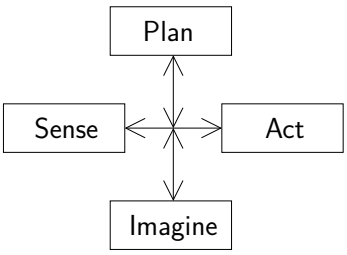


Fig. 5.5: Hybrid Sense-Plan-Imagine-Act robot control paradigm

5.2 Functional Imagination System Architecture

The overall RACE architecture was introduced in Chapter 4 and in Fig. 4.1. In Fig 5.6 the previous architecture is extended and refined to include the capability for functional imagination.

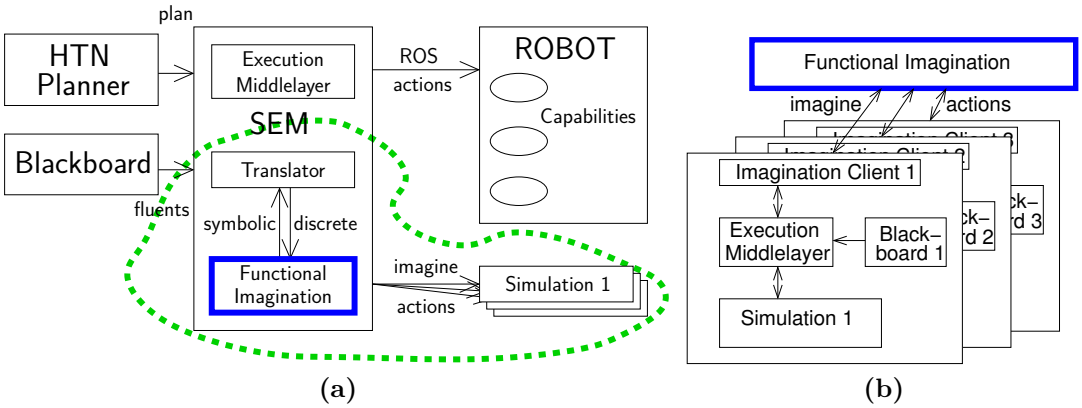


Fig. 5.6: Functional Imagination: (a) Integration of functional imagination (highlighted) into the RACE architecture (excerpt). (b) Detailed view of modules running on external computers.

Before going into module details, it is necessary to describe how functional imagination was integrated into the robot control architecture developed in the RACE[1] project. For greater detail, see [RNZ+13]. An excerpt from this paper that includes the integration of additional modules is presented in Fig. 5.6. All modules query a common store of knowledge called the Blackboard.

[1] http://project-race.eu (March 20, 2015)

Without imagination, the nominal system workflow is as follows. On receiving a new task, the task planner generates a plan based on the current state retrieved from the Blackboard. This plan, which uses the basic capabilities of the PR2 robot to achieve the task, is sent to the Semantic Execution Monitor (SEM). The SEM monitors the preconditions and effects of plan actions and dispatches new actions as appropriate. It maintains a temporal network that represents temporal expectations about execution [KSPS14], e.g. the earliest possible start or finish time of an action. This temporal network is updated based upon real world observations. Temporal observations, such as trajectory durations, could also be obtained by simulation, i.e. from functional imagination.

Actions are dispatched by sending them to the Execution Middlelayer, which calls the appropriate ROS actions implementing basic robot capabilities.

To enable the use of functional imagination, this workflow needed several extensions. Some plan actions need to be simulated with a variety of different parameters before they are executed on the real robot. The plan therefore needs to indicate which actions should be simulated, at which step this should be done and which parameters should be tested. To this end a dedicated *imagine* operator is introduced into the planning domain.

Beside the planning domain, no changes to the planner itself are necessary; this work uses the off-the-shell HTN planner SHOP2 [NIK+03] integrated into the RACE system. Domain modeling of the planner is described in [SGH14]. The new operator `!imagine ?action ?arg1 ?arg2` takes as arguments the action to be simulated and that action's own arguments. Methods using the new operator have to be modified by adding a new action (with arguments): `!imagine !move_base_param ?area slow/fast`. With the action `!move_base_param` the robot is able to drive to a goal position with a specified motion and acceleration set, such as `slow` or `fast`. The definitions for both sets used in the experiments are listed in Table 5.1. The `!imagine` operator can simulate a variety of robot actions. Whereas the `!imagine` operator is specific to simulation, actions like `!move_base_param` apply both to the real robot and to the simulation.

The SEM has to analyze the plan and send each action to be tested to the Imagination Client instead of to the Execution Middlelayer. Furthermore, it is important to note that as the goal is to simulate the execution of an action at a later stage of the plan, the current states of the robot and its environment will be unsuitable. Therefore, the SEM needs to project the state at the later plan stage and send that to the Imagination Client.

Furthermore, the Imagination Client needs to know the poses of all objects and of the robot together with the robot's configuration, in order to initialize the simulation accordingly. This is done by the Translator module, which is given the projected symbolic world state and returns this state in a format that can be used as input to the simulation. This information contains discrete values for the object poses, such

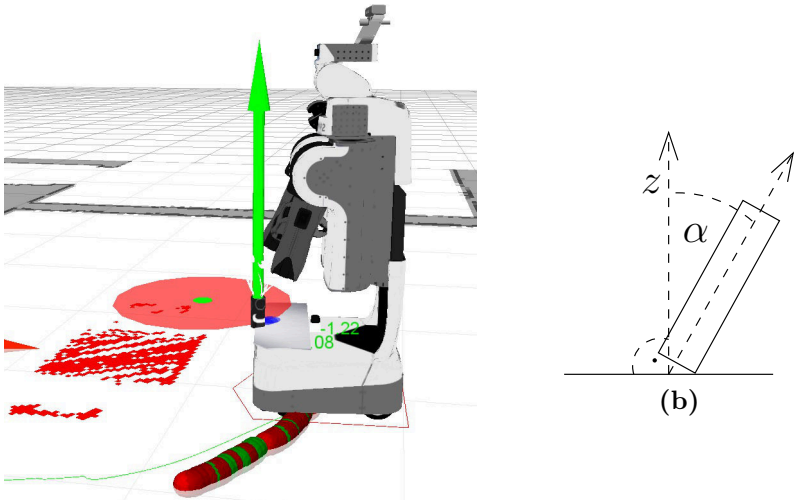
as x, y, z coordinates instead of 'table1'. The translation is based on initial knowledge about a sub-set of the world's objects present in the Blackboard. The SEM generates the expected world state and calls the Translator in order to decouple the Imagination Client from the symbolic representation used by the Execution Monitor. This decoupling allows imagination to be integrated more easily into other systems by adjusting the Translator module. Generation of the expected world state works as follows. For each action `a` given to an `!imagine` operator, a world state has to be generated. The generator considers both the current Blackboard state and the sequence of actions executed before `a`. With this information it is possible to conclude how the world state will have changed by the time `a` is executed.

The Imagination Client will be provided with the altered world state, action to be simulated and possible parameters. At the same time all other operators, with the exception of `!imagine` operators, are executed as usual. Potential further `!imagine` operators are handled the same way as the first. Upon successful execution the Imagination Client returns an ordered list of action parametrizations along with the expected (simulated) duration of the parametrization with the highest confidence. The confidence calculation is described in Section 6.3. If the Imagination Client does not finish simulating an action in time, the plan can wait for imagination to return. However, where there are hard real-time requirements, the action must be executed directly, without waiting, using default parameters defined in the planning domain.

For each parameter combination the Imagination Client starts a separate subset of the RACE system on a different machine, i.e. each with an independent ROS system, as presented in Fig. 5.6.

The prediction system is based on the previously developed HIRES framework, described in [RKZZ14]. It has been integrated into the RACE framework and extended to predict commonsense physics events and to handle uncertainty. A standard PR2 platform equipped with a tray, which is used to deliver a pepper mill, is employed.

In the following paragraphs, model- and scenario-related definitions are introduced. α is the smallest angle between the perpendicular up-axis (z) of the global reference frame and the principal axis of the tray object, as shown in Fig. 5.7. A topple-state s_t is a value from a set of three possible topple-states, $s_t : \{notopple, topple, shaking\}$ which are defined in Eq. 5.1. The thresholds are defined as follows: $\alpha_n = 0.01\,rad$ ($\approx 0.5°$), $\alpha_t = 0.5\,rad$ ($\approx 29°$). The threshold angles were defined empirically and are automatically and continuously measured during experiments. Furthermore any toppling is automatically detected and the tray object is put back on its start position to detect more potential topple events. For the experimental scenario a cylindrical object (the pepper mill) of height $27.4cm$ and radius $3.7cm$ with a point mass of $140g$ has been modelled to match the real object. The object inertia has been defined appropriately.



(a) (b)

Fig. 5.7: PR2 carrying a tray object: The red circle indicates the actual rotation center and radius of the object and the green vector marker indicates the actual up-orientation and reference of the object (a). The object deviation angle α (b).

$$x = \begin{cases} notopple, & if\ \alpha \leq \alpha_n \\ topple, & if\ \alpha \geq \alpha_t \\ shaking, & if\ \alpha_n < \alpha < \alpha_t \end{cases} \tag{5.1}$$

To execute the scenario, two sets of motion parameters θ_{fast} and θ_{slow} were defined, as shown in Table 5.1. The parameter-set elements are necessary for the local motion planner. The individual elements are defined by the standard values for $fast$ and by empirical minimum values for $slow$. The latter were chosen to allow a maximal difference from the $fast$ set while still allowing the local path planning algorithm and motion controller to execute a trajectory smoothly.

	θ_{fast}	θ_{fast}
v_{linear} $[m{\cdot}s^{-1}]$	0.55	0.30
$v_{angular}$ $[rad{\cdot}s^{-1}]$	1.00	1.00
a_{linear} $[m{\cdot}s^{-2}]$	2.50	1.40
$a_{angular}$ $[rad{\cdot}s^{-2}]$	3.20	2.00

Table 5.1: Experimental results: motion parameter sets (θ).

Experiments performed with this approach are discussed in Chapter 6.

5.3 Integration of the PR2 with the Gazebo Simulation

This section elaborates some different approaches to combining a ROS based robot simultaneously with its simulated version. This is not supported by ROS, which has a central node structure that can have only one master node. A related approach with the Player Project framework is presented in [RKZ12]. Here, multiple robots interact and communicate with each other in an agent-based system.

The ROS framework, described in Section 4.2, is used. There are two possibilities to achieve the goal:

1. Multiple networked computers run separate instances of ROS (each with its own 'rosmaster' process). This restricts communication between ROS instances to a proprietary protocol, as rosmasters can only communicate with nodes, not with other rosmasters.

This approach is promising because individual instances run with standard ROS components and configurations. This allows out-of-the-box support for existing robot capabilities, such as navigation, manipulation, vision etc. The drawback is that communication between ROS instances has to be implemented from scratch to transport existing ROS message formats between instances via the network. One approach would be to enable inter-computer communication using sockets. In each ROS instance, a node would listen/publish information on the communication channel within its own "ROS world" with a unique instance Identifier (ID) added. This approach can be implemented quickly for a basic demo (e.g. involving sharing just odometry data) but complexity increases quickly as more information is shared, e.g. for navigation, manipulation etc. Furthermore, being proprietary and thus not (yet) supported by the open-source community, it might become obsolete in future releases of ROS.

2. A second approach is to use existing ROS network facilities. This involves having just one rosmaster running on the network (computers) and sharing information strictly via ROS topics. This method seems straightforward and appealing as it uses the existing ROS interfaces. Nevertheless, at the time of writing, ROS PR2 support is limited to only one PR2 running at a time. Even if this issue is solved (work is ongoing to support multiple simultaneous PR2s), other issues may arise. For example, if different namespaces are used for each robot, existing ROS modules for basic robot capabilities (navigation, manipulation or movement of the robot base) that rely on the standard PR2/ROS configuration (with respect to topic names and namespaces as well as service and parameter names) must be adapted. This tedious work is expensive in developer costs.

In this work the first approach was chosen for following reasons: Namespace issues alone preclude the use of the second approach, but the approach has a further problem in that it violates a ROS requirement that the real and simulated (in Gazebo) robots publish consistent system clock values. In the second approach they would not, causing

problems in all modules. The selected first approach is based on the *multimaster* package[2]. The package provides topics that each rosmaster can access. With this basic facility it is possible to construct any message-based ROS communication between multiple rosmasters.

5.4 Creating an Optimized Simulation Environment

This section describes the creation of an accurate and computationally efficient simulation environment and the related tools used for this work. As described in the previous chapter, Gazebo serves as the simulation base. The following section firstly describes how to build the environment and tools and then presents the procedure for the optimization of simulation speed.

5.4.1 Tools

The basis for an accurate 3D simulation environment is the 2D map created by the robot itself, as described in Chapter 3. The map indicates the footprints of walls and static objects. The following list contains the tools used to create these 3D objects and to adapt properties such as appearance:

- *Sweet Home 3D*[3](GNU General Public License): a free interior design application for drawing the plan of a house, arranging furniture and rendering the results in 3D.
- *Wings3D*[4](Custom License): a lightweight 3D modeling tool.
- *MeshLab*[5](GNU General Public License): an editing tool for unstructured 3D triangular meshes.
- *Blender*[6](GNU General Public License): a comprehensive 3D modeling tool for rendering, animations and game play.
- *Gimp*[7](GNU General Public License): the comprehensive GNU image manipulation program.

[2] http://wiki.ros.org/multimaster (June 7, 2015)
[3] http://www.sweethome3d.com (March 18, 2015)
[4] http://www.wings3d.com (March 18, 2015)
[5] http://meshlab.sourceforge.net (March 18, 2015)
[6] http://www.blender.org (March 18, 2015)
[7] http://www.gimp.org (March 18, 2015)

5.4.2 Performance Requirements and Implementation

With the tools presented it is possible to create the functional simulation environment shown in Fig. 5.8. The use of simulation as a companion tool for real robot execution imposes real-time requirements. The initial simulation setup produced an approximate real-time factor of ~0.1, i.e. 10% of real-time, when running the Blackboard, localization, navigation and manipulation. As such poor simulation performance would not be sufficient in a practical system, several performance improvements had to be made.

This section describes an iterative optimization process that reduces CPU time for simulation. Measurements refer to the real-time factor of simulation (Equation 5.2), i.e. how fast "simulated time" can be computed in simulation. When simulation time passes more slowly than real time, this factor is typically a value smaller than 1.0; when simulation time passes more quickly, the factor is greater than 1.0. Measurements were taken both before and after optimization. The factor F_{sr} represents the simulation time compared with real-time and is defined in Equation 5.2. The time durations T are independent of the task performed, thus T_s and T_r represent the time passed (in s) in simulation and reality respectively. An example would be that the simulation was able to simulate 6 seconds in “simulation time” whereas in real-time 10 seconds passed. This would then result in $F_{sr} = \frac{6s}{10s} = 0.6$. Compared to real-time, the simulation time might not be linear, as the duration of the simulation steps calculation is dependent on the particular situation, e.g. checking for object collisions with one or many contact points. The Gazebo internal physics update rate is defined by the real-time factor F_{rt} resulting from the real-time update rate R_r in s^{-1} and the maximum step size S in s, typically $0.001s$, as shown in Equation 5.3. As this equation is dependent on the computation speed it defines only the maximum real-time factor. A factor $F_{rt} = 1000s^{-1} \cdot 0.001s$ would define a maximum real-time factor of 1.0. Whereas F_{rt} defines the upper limit, F_{sr} measures the actual real-time factor and is used in the following benchmarks.

$$F_{sr} = \frac{T_s}{T_r} \tag{5.2}$$

$$F_{rt} = R_r \cdot S \tag{5.3}$$

Running scenarios with the PR2 in the simulation takes quite a lot of computation time, even on fast computers. For example, a serve-a-coffee scenario (see [ZRP+12] for details) involves 3D collision checking, 2D navigation and collision checking as well as running the RACE framework. This configuration runs at approximately 0.3 times real-time[8]. It should be possible to save some CPU cycles by reducing (appropriately)

[8] The computer used was a 64-bit Intel® Core™ i5-3570 CPU operating at its maximum speed, 3.4 GHz. The CPU had 4 physical cores (without hyper-threading) and 8 GB of working memory. The graphics card was an nVidia® Quadro® 600 with 1 GB of working memory.

the level of environmental detail. Another solution could be to provide a "headless" run of the scenario (without camera sensors).

In the following paragraphs a step-by-step optimization of the simulation environment is given with empirical performance evaluations. In each case, optimization started with an initial setup in which the PR2 robot is placed in the simulation along with the basic wall layout of the TAMS laboratory. The environment is shown in Fig. 5.8. With this reference setup the performance is approximately 0.5 that of real-time.

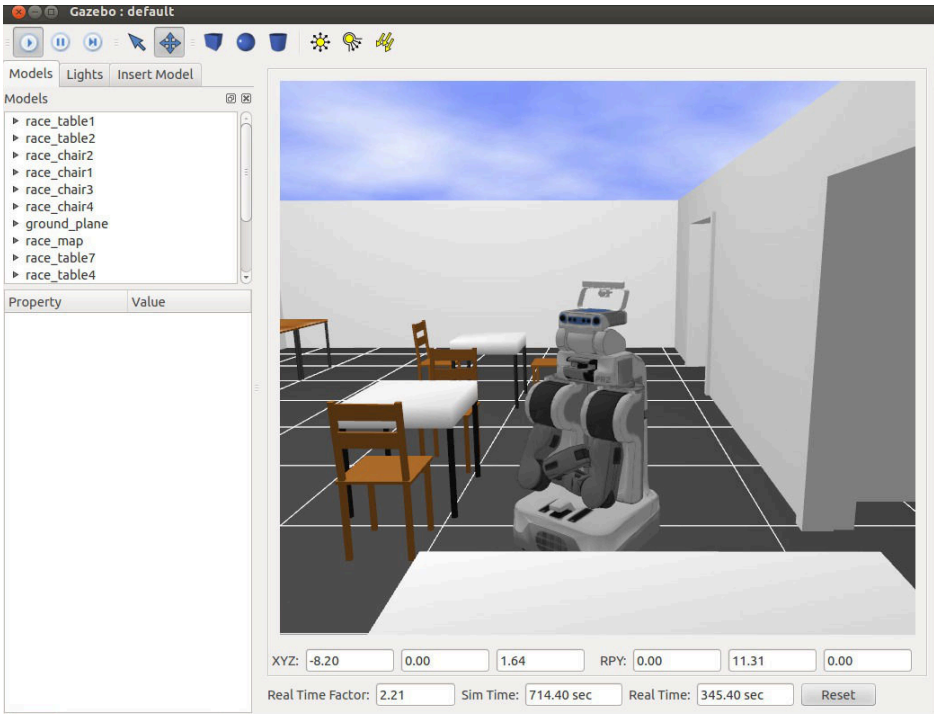


Fig. 5.8: Restaurant environment in Gazebo 1.0 (Fuerte) with a "Real Time Factor" of 2.21

Step 1: Using Static Models. In Gazebo, static objects are not updated during calculation of the next simulation state. This is not the case with dynamic models, i.e. models where joints actually move. Thus changing the `race_chair1.model` links to `static` increases the simulation performance from 0.73 to 0.85.

Step 2: Skip rendering. Adding the `gui` argument to the main launch files allows the simulation to run without the Gazebo GUI. This improves the simulation speed by about 7%. Just launch e.g.
`roslaunch plan_executors_common_scheduler.launch gui:=false`

Step 3: Use simple geometric shapes for collision models. This improves performance in an idle RACE world (tables, cups, robot etc.) from ~0.55 to ~1.50 by optimizing object collision models, especially complex meshes such as those of the cups (Fig. 5.9).

Fig. 5.9: RACE cup mesh visual model (left) and cylindrical collision model overlaid (right)

Furthermore, the reduction in the number of particles in the ROS navigational component of the AMCL node leads to a performance improvement of ~0.1. Nevertheless, navigation (including the `collider_node` for `arm_navigation` that checks for collisions of the arm with the environment) is CPU intensive.

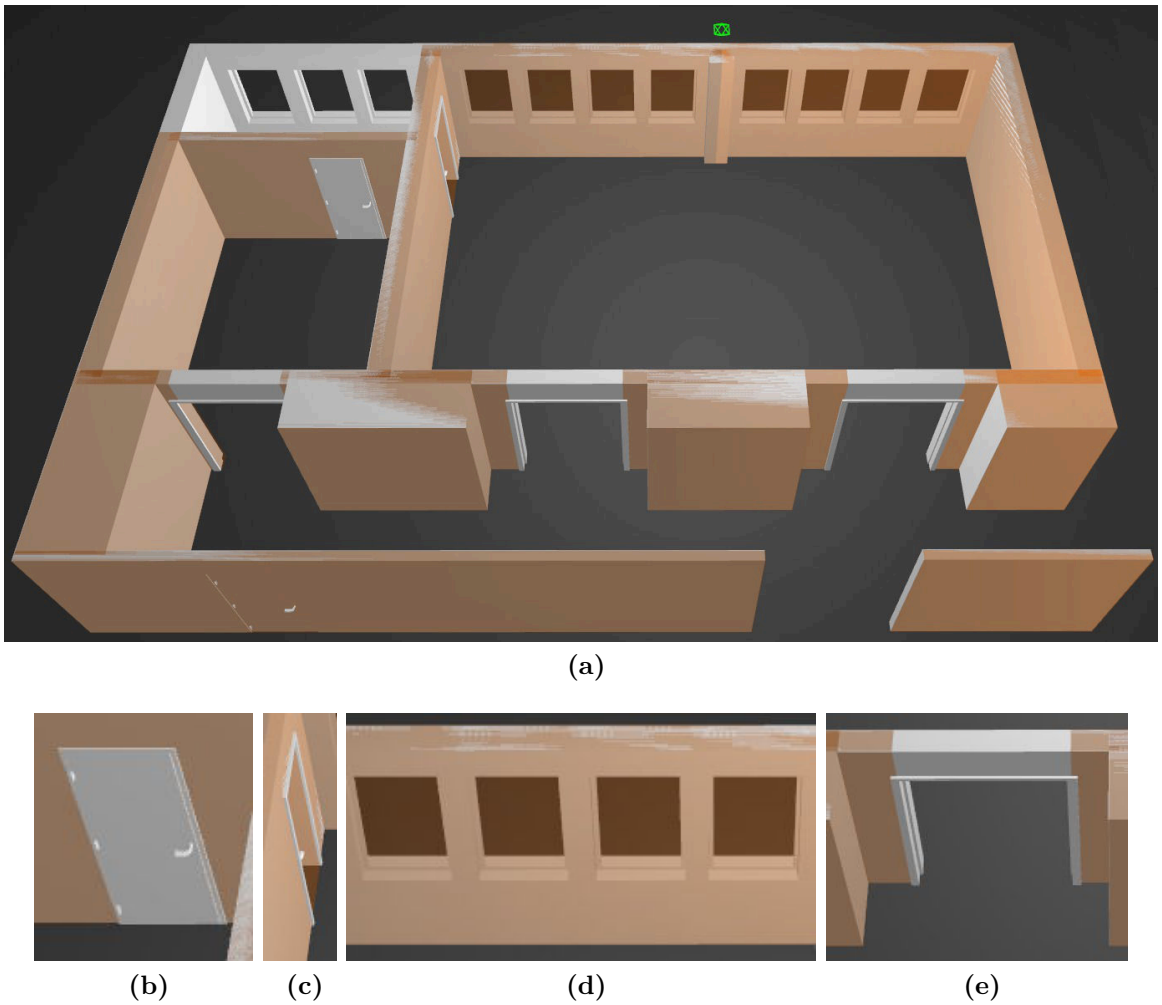
Gazebo runs in a single thread and thus does not currently benefit from multiple cores. A faster single core does lead to improvement.

Step 4: Use simple geometric shapes for wall collision models. This reduces the collision model to geometric primitives (boxes) of background objects (walls, tables etc.) and chairs. Introducing a simplified wall model (for collision only) that contains only flat walls (and holes), no windows etc. improved collision checking performance. These changes improved performance by more than 2.0 (with tables, cups, robot etc.). See also Fig. 5.10.

Several aspects of running a Gazebo world have a major impact on performance. Even things not directly related to simulation, such as navigation and manipulation, can consume many CPU cycles.

Most (if not all) objects are 'rigid'; links in the model (e.g. table legs) do not move with respect to other links in the same model and thus do not have to be updated. This is reflected by setting the `static` tag in a Gazebo model to `true` (once for each model). This leads to a potential performance improvement of ~0.15 (depending on the number of Gazebo objects, such as tables).

Watching the simulation GUI is interesting and sometimes necessary for supervision. If the GUI is not needed, it can be disabled in the launch file (supervising in ROS visualization (Rviz) instead). If the GUI is thus disabled, it can later be launched without



(a)

(b) (c) (d) (e)

Fig. 5.10: (a) RACE walls visual model (gray) and collision model overlaid (orange). The collision model skips fancy features of the visual model for the sake of performance, e.g. the area top-left is unreachable by the robot and is not part of the collision model. (b, c) complex features like the door latch or even the door itself may be omitted. (d) windows too. (e) and door frames.

restarting the whole simulation (by the `rosrun gazebo gui` command). Disabling the GUI yields a possible improvement of ~0.10.

A lot of computation is needed for collision avoidance in the simulation and so using simplified collision models is crucial. Simplification involves using geometric primitives like boxes and cylinders for the collision model (or parts of it) instead of more accurate shapes. This is especially helpful when using a complex mesh (with many faces). For example a wonderful mesh cup could have a cylinder collision model. Such simplifications could lead to an improvement of up to ~2.5.

On the subject of simplification, collisions with walls must almost always be checked. Simplified environmental and collision models therefore help a lot (especially when using a mesh model, e.g. Collada or STL format). Improvements of between ~0.33 and 3.0 are possible.

In Gazebo these improvements can be realized using different models for the `<visual>` and `<collision>` tags. For visualization a sophisticated version can be used and for collision checking a much simpler version might be sufficient. It is irrelevant whether `mesh`, `box` or `cylinder` is used.

Besides the simulation environment, optimization of robot capabilities is also possible. Navigation and localization can easily reduce performance by ~0.5. One can therefore try to reduce the number of particles on which AMCL localization is based, for example, inside the `amcl_node{*}.xml` configuration file. In this work the `min_particles` and `max_particles` are reduced by a factor of 50 (from 500 to 10 and from 5000 to 100 respectively). The navigation system must be verified with these settings. A very accurate localization map created within the simulation itself improves accuracy significantly. An empirical test showed improvements of ~0.10. On the system tested, this is a small improvement. On other systems this number can vary and the optimization is given here only for completeness.

The performance improvements were collected in chronological optimization order and are probably very dependent on the Gazebo world used.

5.5 Modeling Object Properties

Inertia I describes how an object resists rotational acceleration. It can be seen as the rotational analogue of mass[9]. I has units of $[mass] \cdot [length]^2$. In this work and for convenience, objects are considered to have symmetric mass distributions, with constant density throughout the object; the axis of rotation is taken to be through the center of mass. In the case of a cylinder, as shown in Fig. 5.11, the inertia for the central axis, i.e. the z-axis here, is $I_z = \frac{1}{2}mr^2$. The inertia for the x- and y-axes is

[9] http://en.wikipedia.org/wiki/List_of_moments_of_inertia (March 18, 2015)

$I_x = I_y = \frac{1}{12}m(3r^2 + h^2)$. For the cylindric object used in the pepper mill toppling scenario, the resulting Gazebo model description would be as shown in Appendix A.2. This model description contains the inertia and mass definition used to calculate the model dynamics among other things. In this specific example the moments of inertia are scalar, but in general the moment of inertia is a tensor. To obtain the scalar values above, the tensor moment is projected along an axis. The three dimensional inertia tensor for the above example would be as shown in Equation 5.4. The inertia of other geometric shapes can be obtained here[9].

$$I = \begin{bmatrix} \frac{1}{12}m(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \tag{5.4}$$

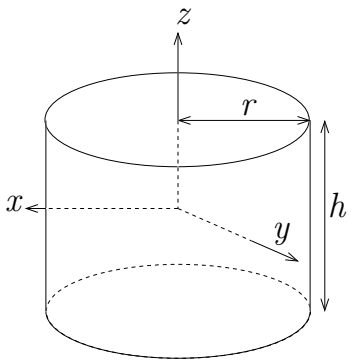


Fig. 5.11: Solid cylinder

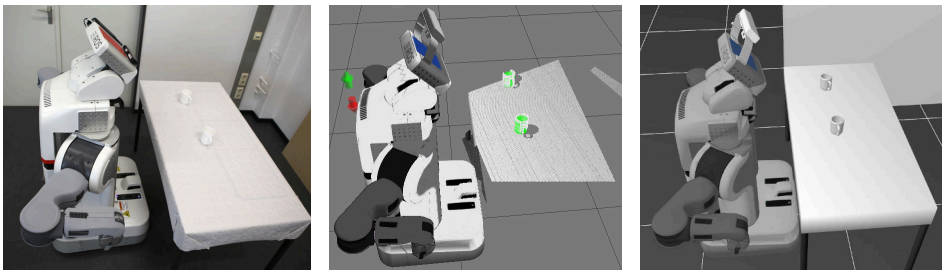
5.6 Update Simulation by Object Recognition Results

Hyperreality is an abstract concept based upon a tight relationship between reality and virtual reality (the simulation). A fundamental aspect of the simulation is that it is updated whenever objects appear or change (in the robot's view) in reality. In this thesis a simulated virtual environment will be used for reasoning about robot action results. This environment will therefore be partially updated with objects the robot encounters in reality. However, no single object recognition algorithm is perfect. Uncertain perception information is always present and must be dealt with. A robust object recognition approach using multiple complementary detectors, as used in this work, is presented in [Kli15].

In a real scenario in which the robot faces a table-top situation, the robot may not successfully recognize the objects on the table. Traditional 3D object recognition involves comparing a segmented RGB-D point cloud against an ideal 3D model stored

in a database. The output of successful object recognition is the object ID corresponding to the correct real world object. In this approach the database is extended with new attributes that store additional information about objects. This information is needed in order to create an instance of the object in the simulation. Objects in the database are referenced by their ID and include at least a *Label* and a *Category*. For Gazebo simulation compliance, the attributes *3D model* and *Gazebo description* were added. The 3D model holds a file name referring to a Collada[10] model that is used for object recognition and for spawning into Gazebo. The XML Gazebo description represents the model information in either the Simulation Description Format (SDF) or the Unified Robot Description Format (URDF)[11] such that it can be spawned into the Gazebo simulation.

Successful object recognition provides the object's ID and also provides its 3D position (and orientation), which is transformed into the Gazebo world coordinate system and used to spawn the object at the correct location. Fig. 5.12 shows the robot situated in front of a counter, recognizing the two mugs on top of the counter; the spawned objects are shown in the simulation. When re-creating a real situation in the simulation, the robot's position is also transformed after it has been retrieved from an AMCL. Any pose applied from the simulation into reality is transformed in the opposite direction. The applied affine transformation is dependent upon the robot's AMCL map relation to the Gazebo world and has to be calculated once for the environment setup in use.



(a) **(b)** **(c)**

Fig. 5.12: A real table-top scenario in which the robot faces two mugs on the table (a). The objects are recognized in reality (b) and are spawned into the simulated scene concurrently (c).

A related area of increasing motivation and effort is the field of mixed reality simulation, which was studied for mobile robots in [CMW09] and in [RKZ12].

[10] https://www.khronos.org/collada (September 8, 2015)
[11] http://wiki.ros.org/urdf (March 16, 2015)

5.7 Generating High-Level Activity Experiences

In this section, a framework for gathering experiences through the process of imagination is described. The framework was developed as part of this work and is based on the Gazebo simulator. The part developed is a software layer to:

- start and stop the Gazebo simulator and the RACE system;
- manipulate the initial robot experience automatically;
- store scenario experiences ordered in episodes[12] on the Blackboard.

In the following paragraphs, two approaches to generating virtual experiences are described. Both feature HIRES and are documented development steps towards the final HIRES system.

5.7.1 Creating experiences with prediction based on robot imagination

Imagination provides prediction of the results of robot actions and activities. This leads to improved robot behavior by detecting successful and failed robot actions - and thus allows replanning - before plan execution fails on the physical robot. The main concept is described in [SHL+13], where a more technical report is given.

In order to integrate the facility to create and use simulations online, i.e. when the RACE system is running, a new component, the 'Mediator', has been introduced. The Mediator communicates with the Blackboard. Its interface allows execution of robot instructions and delivers the results of those executions. These results might then be used by a planner to replan activities. An example would be the currently implemented SHOP2 planner, but future planners, such as a spatial planner, will be supported as well. An overview of how the related modules communicate with the current RACE system is given in Fig. 5.13.

The Mediator aggregates information from multiple simulations and delivers this information to the RACE system. Furthermore it sets up multiple simulations with the information provided by the RACE system.

The integrated Gazebo simulation provides (dynamic) physics constraints (such as object collisions) and kinematic constraints for the PR2, e.g. while manipulating with its arms or moving its base.

[12] In this work, an episode is a coherent set of temporally and spatially related experiences, typically occurring during one scenario run. Running (or re-running) a scenario causes the creation of a new episode inside the robot's equivalent of an episodic memory [Tul72], i.e. its Blackboard (in contrast to the robot's equivalent of a semantic memory, which is its ontology).

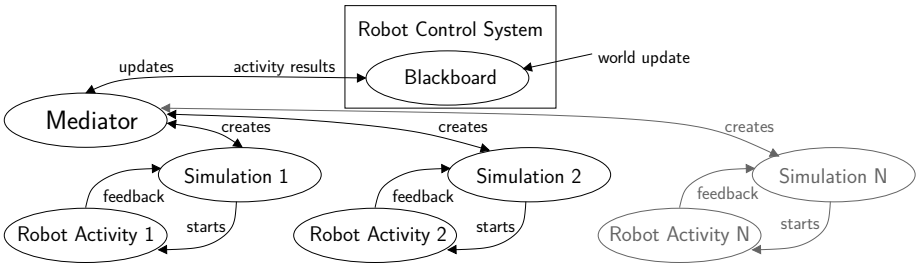


Fig. 5.13: Dynamic View: The Mediator is triggered by the world up-to-date Blackboard. When triggered, it creates a simulation with a defined initial state and asks the robot to execute an activity. This may happen in a loop until the completion criteria are met, e.g. all parameters have been tested.

With this abstract architecture it is possible to develop a system that incorporates multiple simulations to test robot actions (parametrization) ahead of real execution. This approach was refined in Section 5.2.

5.7.2 Unsupervised Experience Acquisition

The collection of a sufficient knowledge base (of experiences) for learning can be a cumbersome task, especially when using a real robot. Whereas the PR2 robot should not operate unsupervised, the simulation of the system can. Therefore a framework is introduced that employs the Gazebo simulation, starts the RACE system, monitors system runs, executes scenarios and takes care of proper shutdown.

The proposed Automated Scenario Execution Framework (ASEF) enables unsupervised learning by creating experiences over many different scenarios. Scenarios can vary, e.g. in the robot's environmental setup and/or the instructions given to the robot. The ASEF creates different scenarios and executes them, recognizing success or failure status automatically. Status can be detected via an activity result that is updated on the Blackboard by the SEM. The latter is described in [SHL+13].

The main benefits are the automatic changes to the robot and its environment in simulation. Furthermore the simulation can rely on accurate sensor values, i.e. without sensor noise (ideal sensors). Fast prototype addition of (as-yet unavailable) sensors is possible.

As an example, a simulated thermal sensor was implemented as a simulated version of the real thermal camera introduced in Chapter 3. It uses RGB color information from objects as a proxy for their thermal properties. A predefined color, such as a hexadecimal RGB value of `0xFF0000` for idealized 'Red', is recognized as a thermal feature. With this approach an ordinary simulated RGB camera interface can be used. A sample visualization is shown in Fig. 5.14.

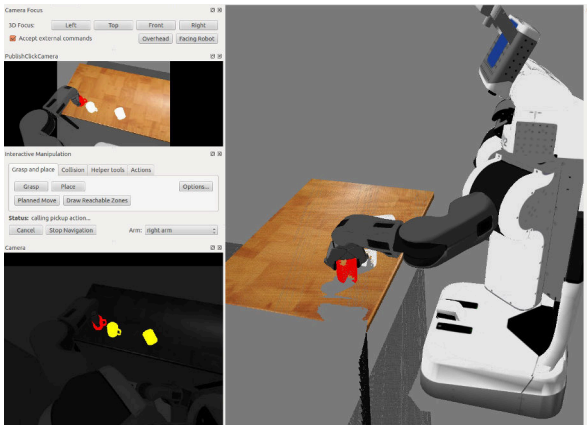


Fig. 5.14: A simulated thermal sensor. The window at the bottom left shows the visualization of mugs with different simulated temperatures (red indicates a higher temperature than yellow).

The ASEF handles the complete RACE system start up (simulation) including robot task initiation. This avoids the need for user interaction and prevents potential brittleness (e.g. of the simulation start up) interfering with scenario execution, i.e. if an error on a (sub-)system might still occur it is detected and the scenario run is aborted and re-started.

A per-scenario configuration file allows the specification of objects in the environment (e.g. tables, chairs and tabletop objects) and of robot instructions. This means that robot instructions, such as "Serve guest1", can be included in the file. An example file is shown in List. 5.1. The format is based on that of a 'fluent'[13] [PMvRH13], as used in RACE, e.g. for loading the initial knowledge into the Blackboard. It provides some convenient features. `Attachments` binds entities together spatially in a hierarchy. This is useful e.g. when moving a table that has chairs 'attached' to it. `Modifications` allow specification of alternative positions for an object. Each coordinate (axis) takes a vector of different values that define the spatial difference between multiple scenario runs.

[13] A fluent is a semantic spatio-temporal data structure that represents current knowledge (i.e. an experience) and is the atomic data type of the Blackboard. It is an instance of a concept from the ontology and usually has a start and finish time.

```
!FluentPoseModification
Instance: poseTable1
Group: testGroup1
Choices: 3
Modifications:
  -  [hasX, [0, 0, 1]]
  -  [hasY, [0, 1.5, -2]]
  -  [hasZ, [-0.375, -0.375, -0.375]]
Attachments: [poseChair1, poseChair2]
---
!FluentPoseModification
Instance: poseTable2
Group: testGroup1
Choices: 3
Modifications:
  -  [hasX, [0, 0, 3]]
  -  [hasY, [0, 1.5, 1]]
  -  [hasZ, [-0.375, -0.375, -0.375]]
Attachments: [poseChair3, poseChair4]
---
!FluentPoseModification
Instance: poseCounter1
Group: testGroup1
Choices: 3
Modifications:
  -  [hasX, [0, 3, -1]]
  -  [hasY, [0, -0.7, 2]]
  -  [hasZ, [-0.375, -0.375, -0.375]]
  -  [hasYaw, [0, -1.57, 0]]
Attachments: [poseMug1, poseMug2]
---
!FluentPoseModification
Instance: posePR21
Group: testGroup2
Choices: 3
Modifications:
  -  [hasX, [0, -1, 0]]
  -  [hasY, [0, 1, 1]]
  -  [hasYaw, [0, -2, -0.43]]
Attachments: []
```

List. 5.1: An example configuration for three different scenario runs.

The framework loads the environmental information directly from the Blackboard (initial and scenario knowledge) and applies user-defined changes in order to create the simulation environment. The resulting environments could look like those depicted in Fig. 5.15, in which three different spatial environments are created in sequence for a coffee serving scenario.

Fig. 5.15: Examples of three scenario runs. Each has a different initial environment setup but the robot was given the same instruction: “Serve a coffee”.

During scenario execution, various information is given to the user (via the terminal), such as the number of fluents currently being processed. After each run a per-run summary is given and on completion of all runs a final summary is output. These summaries include the number of episodes created and their references in the Blackboard, the instructions given to the robot, the time taken and brief statistics on the total number of fluents and the average fluents processed per-second, as shown in List. 5.2. A possible future addition will be the management of different ontologies (updated after learning).

```
$rosrun race_simulation_run sim_run 20
...
= RUN 20 SUMMARY =
=[ INFO]: Scenario episode is: '2013-10-10T20:14:10'
=[ INFO]: Scenario instruction: 'achieve serve_coffee_to_guest_Task guest1'
=[ INFO]: Total (sim) time used: 300.21 seconds
=[ INFO]: Total number of Fluents used: 381
=[ INFO]: Average ratio of (Fluents/sec): 1.27

= TEST SUMMARY =
=[ INFO]: 20 episodes created during test with names: '2013-10-10T16:50:30'
'2013-10-10T17:00:33' '2013-10-10T17:13:53' '2013-10-10T17:27:57'
'2013-10-10T17:41:08' '2013-10-10T17:52:11' '2013-10-10T18:03:46'
'2013-10-10T18:13:40' '2013-10-10T18:25:29' '2013-10-10T18:36:06'
'2013-10-10T18:47:46' '2013-10-10T18:50:54' '2013-10-10T19:05:33'
'2013-10-10T19:19:26' '2013-10-10T19:22:34' '2013-10-10T19:33:12'
'2013-10-10T19:44:05' '2013-10-10T19:47:12' '2013-10-10T20:00:35'
'2013-10-10T20:14:10'
=[ INFO]: Average Fluents per episode: 386.42
=[ INFO]: 37 episodes are now in repository 'sim_run1'
```

List. 5.2: An example output of 20 scenario runs.

5.8 Generating Dynamics-based Experiences

The previous section described an approach in which experience was obtained in an unsupervised way by automatically executing scenarios while significantly changing the robot environment. In contrast, this section describes an approach in which the robot executes scenarios in a known, fixed, environment. Whereas before it was of interest whether the robot could still solve its task with an altered environment, here the question is whether the robot can fulfill its task in a fixed environment. And if not, whether it is possible to learn the cause of failure in order to ensure future success. This work focuses on providing the framework and data for learning. Actual learning is not included here.

Learning from toppling events requires many scenario runs. Simulation is an ideal tool to obtain the necessary training data, as it allows direct object tracking and creates reproducible situations. The HIRES system [RKZZ14] is used as the underlying framework. All of the resulting data is useful: toppling events (f_{topple}), where a pepper mill (or any other object carried on the robot's tray) actually topples; shaking events, where the pepper mill bounces but does not topple; and non-toppling events (no bouncing). Furthermore the robot dynamics (R_{dyn}), such as velocity v, acceleration a and jerk j, should be included in the training data, as should the dynamics of the pepper mill (O_{dyn}). Dynamics data also includes the current position $P_{xy\phi}$ in 2D (x, y, ϕ) at time t. Definitions and relations between an episode E (set of fluents) and a fluent f (n-tuple of time, episode id, object deviation, object dynamics and robot dynamics) are shown in Equation 5.5 and 5.6. The data visualization is shown in Figures 5.16, 5.17, 5.7, 5.18. In the work presented, a vase substitutes for the pepper mill as a tall tray object.

$$
\begin{aligned}
f &\in E \\
\forall f \in E, \quad & \exists f_{topple} = 1 \\
f &= (t, E_{id}, O_{dev}, O_{dyn}, R_{dyn}) \\
O_{dev} &= (\alpha, s) \\
O_{dyn} = R_{dyn} &= (D_{lin}, D_{ang}, P_{xy\phi}) \\
D_{lin} = D_{ang} &= (v, a, j) \\
P_{xy\phi} &= (x, y, \phi)
\end{aligned}
\tag{5.5}
$$

$$
\begin{aligned}
\vec{s} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}, \ \vec{\omega} &= \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix} \\
v(t) &= \frac{s(t_2) - s(t_1)}{t_2 - t_1} \\
j(t) &= \dot{a}(t) = \ddot{v}(t) \\
\omega(t) &= \frac{\alpha(t_2) - \alpha(t_1)}{t_2 - t_1} \\
a_{angular} &= \omega(t) \\
j_{angular} &= \dot{a}_{angular}(t) = \ddot{\omega}(t)
\end{aligned}
\tag{5.6}
$$

In this scenario, an episode starts from the beginning of the scenario or from the last toppling event and lasts until the next toppling event. During an episode, fluents containing significant information are constantly created and stored. The fluent high-level format is shown in Listing 5.3 and a sample fluent is shown in Listing 5.4 in a data-mining friendly format.

To train with realistic data, a scenario was created that lets the PR2 robot drive to a pre-manipulation area (PMA) selected sequentially from those stored in the Blackboard. A qualitative comparison of real and simulated results is presented in Chapter 6, Section 6.1.

5.8.1 Toppling Detection

Detection of toppling is generally straightforward in a simulation. Every object is constantly updated in the simulation event loop - at $1000Hz$ in Gazebo. An object's position on the robot tray can be queried and can be used to calculate quite accurate object dynamics. In this scenario an object like a pepper mill is tracked. To obtain a vertical deviation, the object's normal vector is compared to its initial normal vector (robot z-axis, upward), see Figure 5.7. To obtain the horizontal deviation the object's current position is compared to its initial position. Any toppling event is detected and stored in the data trace for later processing. As described before, a toppling event concludes an episode and starts a new one, resetting the tray object's pose and orientation.

5.8.2 Data Storage and Learning

In order to provide a facility for (offline) learning, all data generated during scenario execution is stored on-the-fly in a database. The database can later be mined to aid learning about the toppling demo.

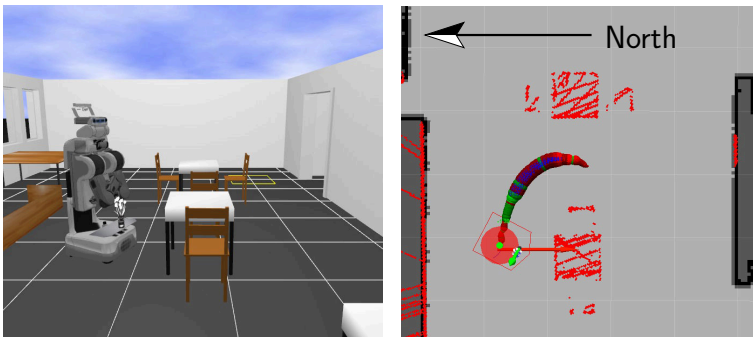
```
# Message to publish base translational velocity, acceleration and jerk
# angular velocity, acceleration, jerk
std_msgs/Header header
uint32 EpisodeID
string state # 'notopple', 'shaking', 'topple'
geometry_msgs/Vector3Stamped obj_dev # Object angular and linear deviation
race_pr2_markers/BaseDynamics obj_dyn # Object absolute dynamics
race_pr2_markers/BaseDynamics base_dyn # Robot absolute dynamics
```

List. 5.3: ToppleEpisodePart high-level fluent message definition

```
57492000000,0,57492000000,,
0,
notopple,
1023,57492000000,/base_footprint,0.000200874365274,0.0146495010202,0.0,
0,57154000000,/map,0.108661428094,0.117334552109,0.693616688251,
  9.20355669223e-05,0.0,0.0,0,0,,0.0,0.0,0.0,
0,56996000000,/map
    ,0.000426712387707,0.00571070192382,0.113538585603,0.370499312878,
0.464733093977,2.83434391022,0,56996000000,/map
    ,7.93312692642,12.701253891,-1.86824214458
```

List. 5.4: Sample fluent for a 'notopple' event

The monitored data visualization is shown in Fig. 5.16, 5.17, 5.7 and 5.18.

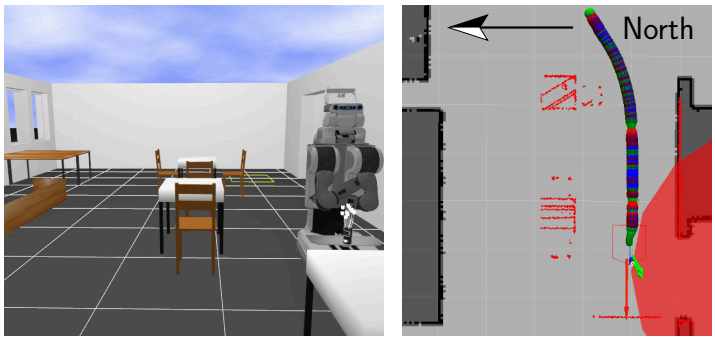


(a) **(b)**

Fig. 5.16: PR2 approaching PMA north of table1 in the simulation (a). The thickness of trace markers indicates speed, colors indicate acceleration. The red circle indicates the actual rotation center and radius of the tray object. The color indicates positive acceleration in green and negative acceleration in red (b).

5.9 Many Worlds Interpretation

When it comes to imagining a fictional world, a logical next step is to extend that approach to imagining many worlds in parallel. Whereas this is difficult for humans



(a) (b)

Fig. 5.17: PR2 approaching PMA east of counter1 in simulation (a): A smooth trajectory allows 'acceleration-sparse' movement (b).

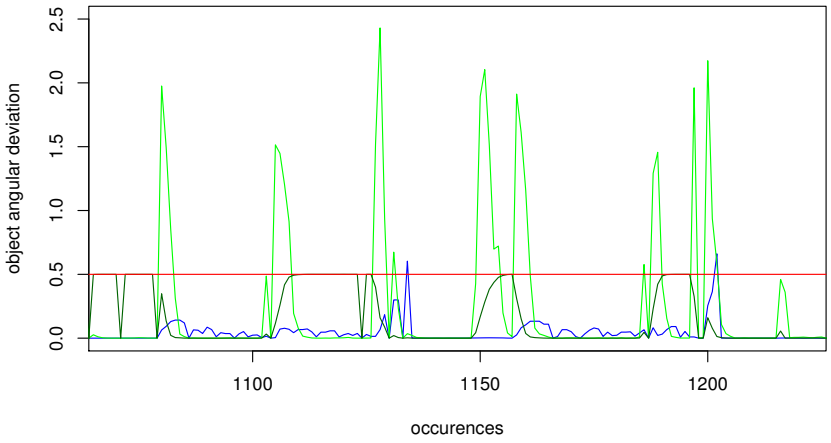


Fig. 5.18: Angular object deviation vs. linear velocity/acceleration: The blue line indicates the object's angular deviation from the vertical reference vector. The dark green line indicates the velocity. The light green line indicates the acceleration. All values are absolute values. The Y-axis indicates angular deviation in radians. The horizontal line indicates the empirical value where the object topples and is reset.

it is straightforward for computers if appropriate resources are available. A system imagining multiple scenes of the same or different scenarios would naturally save time resources in the process of predicting future events.

The Many Worlds Interpretation (MWI) is a theory originating from quantum physics. It states that all possible alternative histories exist in parallel, i.e. any event that can possibly happen does actually happen thus leading to a "many-worlds universe". Hugh Everett III first published this theory in 1956 in his doctoral thesis *The Theory of the Universal Wave Function*, which was compiled later in [BSD73]. It is about the second most accepted theory in quantum mechanics, after the Copenhagen theory [Teg98]. Some recent popular work by David Deutsch employs the concept in theoretical support of quantum computers [Deu85].

In theory, the ability to access multiple worlds allows the results of different actions or action parametrizations to be evaluated. The common ground (i.e. the domain) has to be the same in all worlds investigated. In practice this would work as follows. Firstly, a scenario is created, different actions are applied (with a robot) within the scenario and the results are collected. The results are then evaluated and the best is selected (according to some selection criteria). The "action parametrization" from the selected world is then extracted and applied to the real environment. The term "action parametrization" will be explained later.

In order to predict events, the range of possible alternatives is important. Relevant questions would, for example, be: what would happen if the robot took this route rather than that route, what if it placed an object in one position on the table rather than another? Combining the theory of MWI with the capability of today's available computational power allows a many-simulation interpretation of future action results, in particular for robotics.

One programmatic approach is to map an action parameter to one simulated world. For each reasonable parameter a new world is created, with the upper limit to the number of possible worlds depending upon the available computational power. After an action finishes, the related world is either terminated, in order to save resources, or re-used for a similar simulation. Each simulation provides information on the result of the action and back-tracking leads to the parameters which caused that result.

Another approach is not to limit a simulation to one action but to "keep it running". That means creating a simulation at a specific time, executing consecutive actions within it and tracing the simulation state at the end of a complete episode (or indefinitely). As small variations during the simulation can lead to very different simulation states, this approach is very computationally intensive and thus provides very little practical granularity.

In the example of a robot given the task of moving to a table, depending upon its initial position and its environment, the robot can decide whether to approach the table from one side or another. The decision will determine the robot's trajectory and

might have other implications. The action could be parametrized with two valid goal positions. Thus in this work, the term "action parametrization" refers to the possibility for a robot action to be instantiated with different parameters that possibly result in different world states after execution.

5.10 Summary

This chapter developed the methods and experimental design required in order to ensure a robust realization of the approach initially stated (cf. Section 1.4). The scenario setup was introduced and the required robot capabilities were developed. A model for representing the problem domain was developed and a detailed architecture was created. The chapter also introduced the tools and methods necessary for the creation of a simulation environment for the problem domain. In order to help meet real-time constraints, a detailed performance optimization was performed on various elements of the simulation. A framework proposal to integrate imagination into a Blackboard-based robot control system, such as the RACE framework, was given and two frameworks for creating unsupervised experiences were described. Finally an overview of overlapping aspects of this work and the MWI was presented. The next chapter, Chapter 6, describes experiments that measure the effects of adding functional imagination to a robot system.

Experimental Scenarios and Evaluation

6

This chapter presents the experiments used to evaluate the functional imagination approach described in Chapter 5. The simulation used to implement imagination is verified for accuracy and repeatability. If the simulation is accurate, it is possible to compare simulated and real world action results. Assuming the correctness of the simulation, it will be shown that predicted results from a simulation can prevent failures in reality. Furthermore the results will show that multiple aspects of reasoning (spatial, temporal, causal) derived from simulation can enhance a plan-based system that uses execution monitoring and will thus support the hypothesis stated in Section 1.4.

In order to verify simulation the same experiments were executed in simulation (Gazebo) and on the real robot (PR2). The experiments for evaluating action result prediction were performed on the real robot (with simulation based imagination). The experimental restaurant environment is shown in the ROS visualization in Fig. 6.1. This figure shows labels of both real world and artificial concepts. Whereas most labels in the robot domain relate to existing objects (for example the label 'table1' refers to the first table in the restaurant environment), a label for an artificial concept has also been introduced. The representation of a pre-manipulation area (PMA) was introduced because the underlying collision avoidance system prevents the robot from navigating close to an object. The robot also has limited ability to manipulate objects. For example, putting an object on the robot's tray would cause a collision with the table next to the robot. With the concept of a PMA, a robot action dedicated to navigating close to tables was developed for the RACE project. Throughout this work, whenever a robot trajectory is mentioned, it refers to a trajectory between PMAs.

The following sections first present the results of validating simulated actions against reality. Subsequently, the results of executing the proposed system on a PR2 are presented.

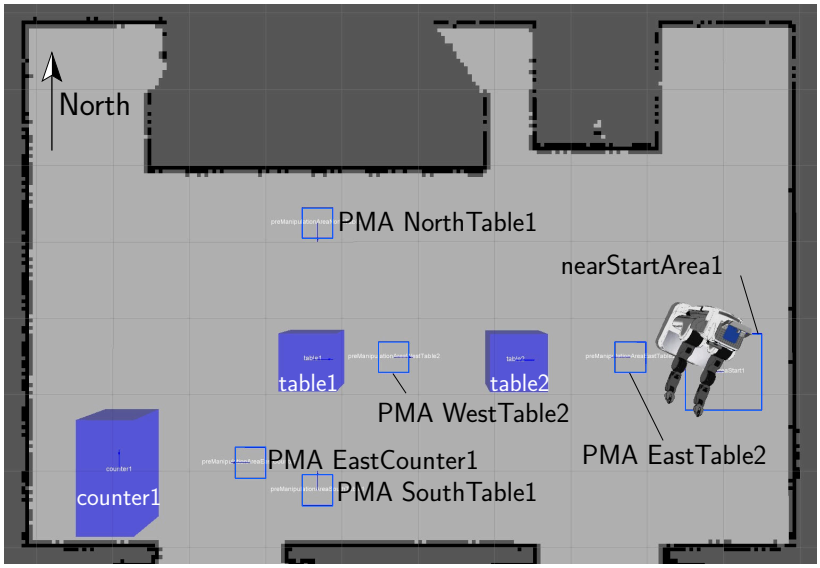


Fig. 6.1: Restaurant floor map with markers for the counter and the two tables (filled blue boxes) and for each PMA (one for a possible approach direction): two for each table on opposite sides, one for the counter and one in the start area near the door on the right. *North* is defined here as upwards.

6.1 Simulation Validation

Is a physics-based simulation deterministic or does it include (some) in-built noise? A simple deterministic scenario is created for an object on a robot tray almost toppling while the robot is in motion. The deviations are measured and compared for several trails. If simulation is deterministic the results should be very similar. The Gazebo simulation employed is deterministic in the sense that repetitions of the same scenario will most likely lead to similar results. This is not guaranteed however, as system behavior is affected by other aspects, such as operating system scheduling. Scheduling alters both the timing of the simulation and the allocation of CPU resources to the calculations. In combination with floating point rounding effects, this can have a minor influence on simulation outcomes.

Verification and validation of computer simulations is very common in the development of accurate and credible models. In this work the simulation model is used to predict real-world states and thus depends strongly on accuracy and reliability. Simulation is widely used in the development of software algorithms for areas such as exploration, mapping and navigation. Two important limitations are that Gazebo, or more precisely the ODE physics-engine, can only simulate rigid-body dynamics and that its monolithic dynamics engine does not allow the use of distributed computation.

As a partial conclusion: OS (Ubuntu Linux) scheduling leads to at least partial randomness. This cannot be changed without altering the Gazebo code base itself, which

is outside the scope of this work.

There has been some work on this topic in the literature. [BB07] presents a qualitative evaluation of free, publicly available, physics engines. Its criteria are the number of constraints implemented (among others). The most useful constraints are models of real life system behavior. Common constraints are, for example, those affecting joints (prismatic, revolute and spherical). [WHR10], in contrast, evaluates the Bullet engine as an ingredient for physical reasoning. It studies how accurately the pushing of flat objects across a table with a robotic arm can be predicted. [OFY07] implements an omnidirectional camera model for Gazebo and verifies it by qualitative comparison of rendered images with images obtained by a real omnidirectional vision sensor.

[KH04] describes the design and usage paradigms of the Gazebo simulator. Fig. 4.2 shows how a model is constructed in a tree-like hierarchical structure and how world models are connected to the external Gazebo interfaces. [FCK+09] verifies a Zigbee based distributed communication system by comparing simulated and real results.

Generally a physics-based simulation has no noise apart from the minor noise introduced by the underlying hardware and OS, i.e. scheduling. An obvious exception is noise that is explicitly modeled, such as the Gaussian noise used in the Gazebo laser simulation. Also, the standard deviation σ of the durations of actions measured in the restaurant environment within Gazebo is small, as can be seen in Table 6.1. The corresponding measurements with the PR2 are shown in Table 6.2. A comparison of the same actions in simulation and in reality is shown graphically in Fig. 6.2. The two tables list the action executed, with parameters as appropriate. In each table, 'Samples' is the number of iterations performed, $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$ is the standard deviation of the duration, μ is the mean duration and 'to/from' gives the acronyms of the starting pose or posture of the robot. As discussed earlier, for the `move_base` action, ET2 corresponds to EastTable2, EC1 is EastCounter1 and so on. For the `move_torso` action, 'U' means up and 'D' means down. The `tuck_arms` action uses 'U' for untucked arms and 'T' for tucked arms.

The probability distributions of these measurements are shown in Appendix A.8. An example measurement is shown in Fig. 6.3. It shows the `!move_base_param` action driving the robot from PMA east of counter1 to PMA south of table1. Although the test data size is rather small with only 24 samples, the histogram and the overlaid density of action duration show an approximately standard Gaussian distribution. The QQ plot shows independent standard normal data (in seconds) on the vertical axis against a standard normal population on the horizontal axis. The linear fit to the data indicates that they are normally distributed. There are some clear outliers that do not fit (five in total in Fig. 6.3 a,b and two in Fig. 6.3 c,d). They show up disproportionately in the plots because of the small sample size.

The validation data was retrieved by running the robot capabilities (perception, actuation) and the RACE system. The system load on each computer performing the

Action	Parameter	to/from	Samples	σ	μ
1 !MOVE_BASE_PARAM	fast	EC1/ET2	20	0.45	14.04
2 !MOVE_BASE_PARAM	slow	EC1/ET2	20	0.59	20.22
3 !MOVE_BASE_PARAM	fast	EC1/NT1	20	0.72	15.99
4 !MOVE_BASE_PARAM	slow	EC1/NT1	20	0.57	18.38
5 !MOVE_BASE_PARAM	fast	ET2/EC1	20	5.38	24.08
6 !MOVE_BASE_PARAM	slow	ET2/EC1	20	0.82	27.00
7 !MOVE_BASE_PARAM	fast	NT1/EC1	20	0.89	15.33
8 !MOVE_BASE_PARAM	slow	NT1/EC1	20	0.89	19.06
9 !MOVE_TORSO		U/D	20	0.02	20.99
10 !MOVE_TORSO		D/U	20	0.04	21.69
11 !TUCK_ARMS		U/T	20	0.09	9.25
12 !TUCK_ARMS		T/U	20	0.08	5.70
Average				0.88	17.65

Table 6.1: Quantitative results of the standard deviation σ of various simulated actions.

Action	Parameter	to/from	Samples	σ	μ
1 !MOVE_BASE_PARAM	fast	EC1/ET2	20	1.57	16.40
2 !MOVE_BASE_PARAM	slow	EC1/ET2	20	1.47	22.59
3 !MOVE_BASE_PARAM	fast	EC1/NT1	20	0.95	16.85
4 !MOVE_BASE_PARAM	slow	EC1/NT1	20	0.53	19.41
5 !MOVE_BASE_PARAM	fast	ET2/EC1	20	1.23	23.07
6 !MOVE_BASE_PARAM	slow	ET2/EC1	20	1.39	29.86
7 !MOVE_BASE_PARAM	fast	NT1/EC1	20	1.07	16.59
8 !MOVE_BASE_PARAM	slow	NT1/EC1	20	0.46	20.89
9 !MOVE_TORSO		U/D	10	0.03	23.39
10 !MOVE_TORSO		D/U	10	0.07	22.19
11 !TUCK_ARMS		U/T	10	0.03	8.58
12 !TUCK_ARMS		T/U	10	0.01	5.29
Average				0.74	18.76

Table 6.2: Quantitative results of the standard deviation σ of the duration of various real robot actions.

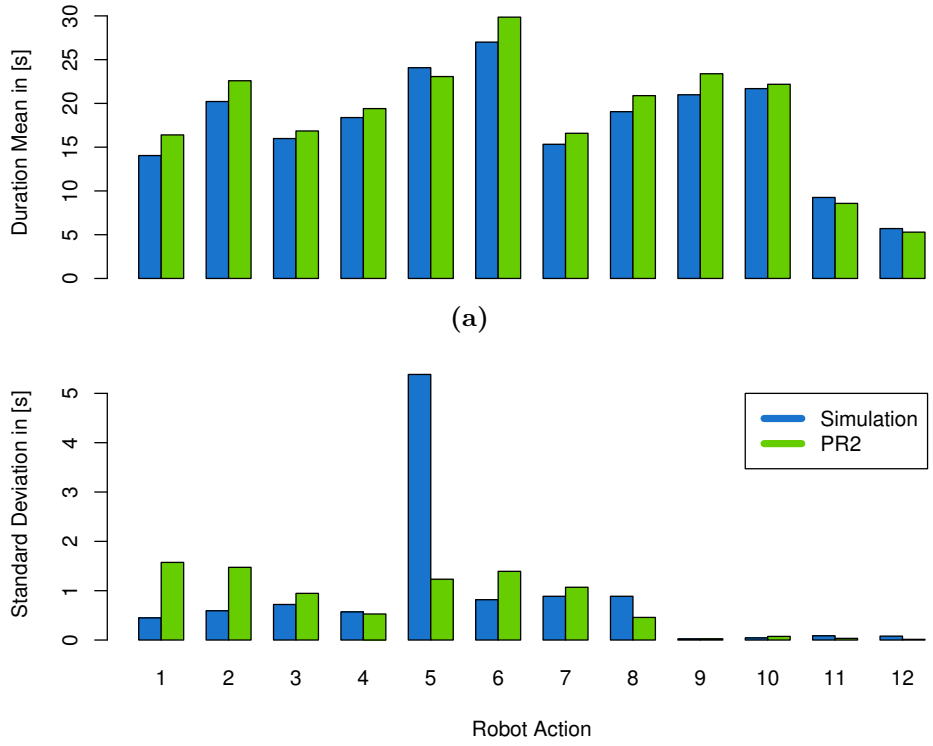


Fig. 6.2: Simulation vs. PR2 action duration. The two charts show the simulation and PR2 action duration mean, μ, (a) and their standard deviation, σ, (b), both in seconds. The actions on the $x - axis$ correspond to those in Tables 6.2 and 6.1

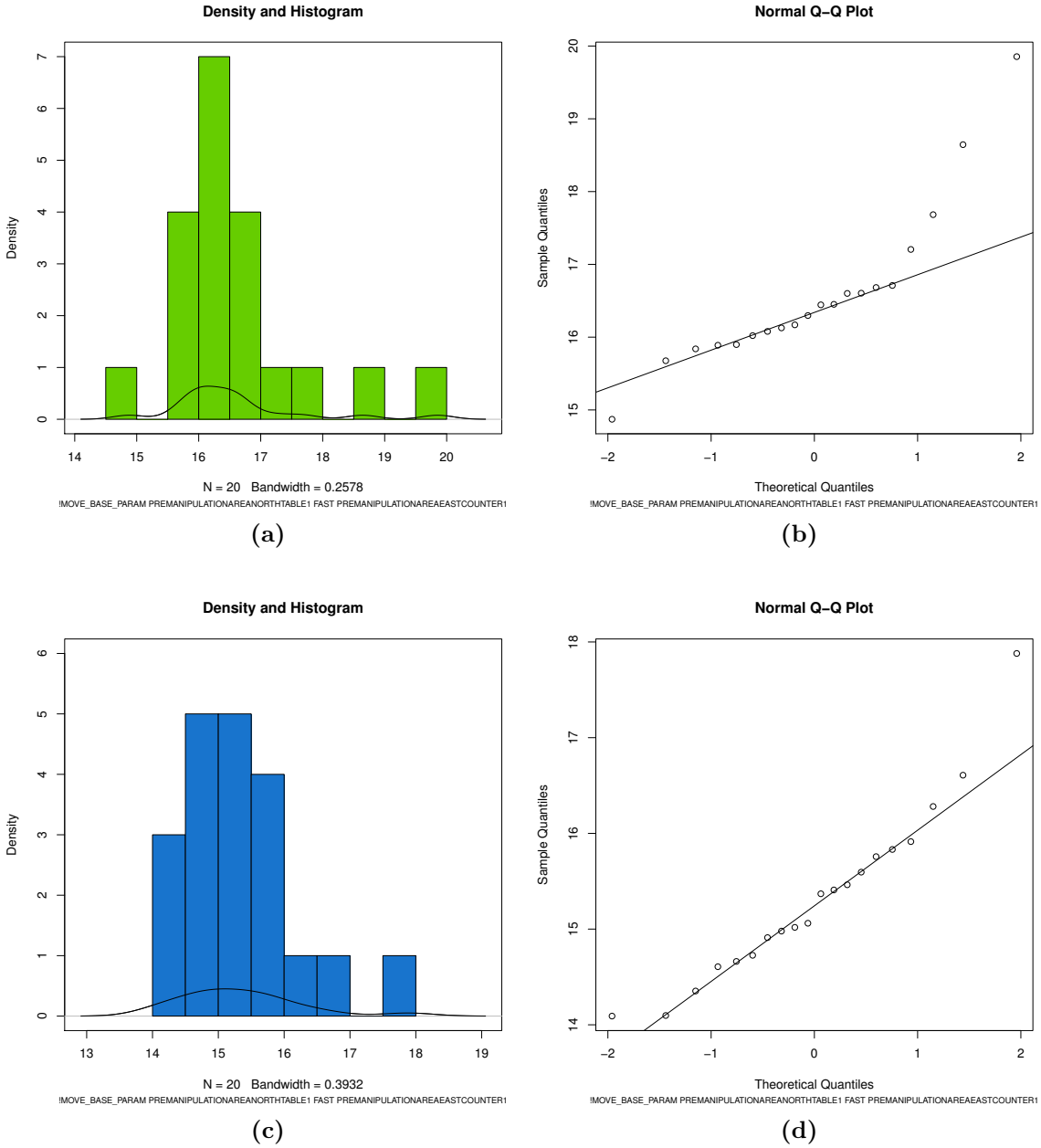


Fig. 6.3: Statistics of action `!MOVE_BASE_PARAM` on the real robot (a,b) and in simulation (c,d). (a,c) Density and Histogram. (b,d) Q-Q Plot.

simulation is close to its limit. System load will thus influence the measurements slightly. The data shows that σ and μ are slightly smaller in simulation than on the real robot and that arm and torso movements (actions 9 to 12) have very small σ both in simulation and in the real environment. Moving the base from ET2 to EC1 (the 5th action) seems to have a very high σ in simulation, whereas the same trajectory in the opposite direction (action 1) has a very low σ ($< 0.5s$). The total duration might differ according to the robot start position. The duration depends heavily upon the robot's orientation when it reaches or leaves the table. The orientation determines whether or not the robot must rotate. Nevertheless the test data reveals three tests with obviously excessive durations ($< 36s$) that indicate a problem. The most likely explanation is that a navigation problem resulting from a loss of localization caused the action to be aborted or led to the selection of an unfavorable choice of low-level navigational behaviour. In summary, the test data qualifies the σ value of action 5 as an outlier.

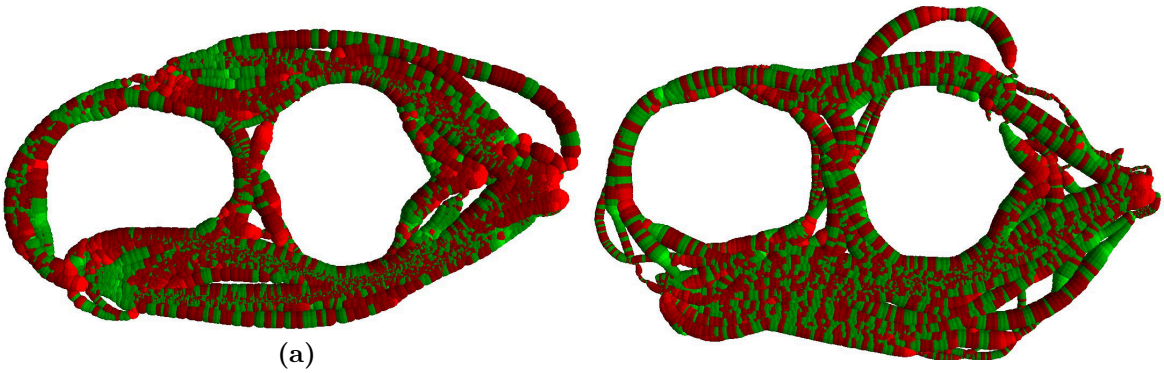
Ignoring outliers, the first observation is that there are only small differences between simulated and real results, with a maximum difference in mean values of $2.9s$ (action 6). The difference may reflect the quality of sensor data, which is almost perfect in simulation but includes significant noise in reality. Odometry and laser sensors, which are used for localization and path planning, are noisy due to wheel slippage and to reflections or absorption of the laser beams by the environment. Furthermore, the laser sensor has inherent noise. The simulated friction and laser noise (which has a Gaussian distribution) may imperfectly model reality. Arm and torso movements depend on real and modeled joints, springs and friction. Deviations could result from slightly inaccurate modeling parameters.

Another qualitative comparison between simulated and real navigation is shown in Fig. 6.4. In this experiment the robot was instructed to approach every PMA in the restaurant. The resulting trajectories are recorded and visualized. The simulation ran for about four hours (simulation time) corresponding to about one hour here in real time. In the figure the thickness of the trace indicates the robot's velocity with a maximum of $0.55ms^{-1}$, green indicates positive acceleration and red indicates negative acceleration. The few outliers from the real trajectory result from a noisy laser scanner adding "ghost obstacles"[1] that cause the path planner to add detours. The two free spots in the trajectory indicate the two restaurant tables, as can be seen in Fig. 6.1. Deviations are mainly caused by sensor noise from the real sensors and by the systematic "ghost obstacle" issue. With those caveats, the two trajectories look very similar, especially the velocity and acceleration readings.

Related diagrams showing velocity against time for both simulation and reality are shown in Appendix A in Fig. A.6. The "ghost obstacle" issue manifests itself in diagram artifacts. Linear and angular density histograms are shown in Fig. A.7. Jerk

[1] http://answers.ros.org/question/37126/pr2-sees-ghost-obstacles-in-tilting-laser (March 16, 2015)

densities in simulation and reality are shown in Fig. A.8. Comparisons of the object dynamics with robot base dynamics are shown in Fig. A.9, A.10, A.11, A.12 and A.13.



(a) **(b)**

Fig. 6.4: Real and simulated robot trajectories compared: Simulation (a), real (b). The robot was instructed to approach every PMA in the restaurant.

6.2 Spatial Prediction

The following two scenarios are introduced to show the spatial reasoning capability of HIRES. In the first scenario the robot evaluates different self-torso positions in combination with four table-relative approach configurations in order to determine the optimal robot configuration for object recognition and for table-top manipulation.

6.2.1 Torso Poses

Table-top manipulation often occurs in robot service tasks in human environments - and often fails. The complexity of such actions includes perceiving the table and objects, planning collision-free arm motion and grasping the object in a stable way. Due to sensor noise and to the robot's position in relation to the table and objects, perception and motion planning can sometimes fail.

A simple experiment illustrates the use of simulation in such situations. Instead of giving the robot a fixed pose in front of the table, it is given four poses: default, left, right and behind. The robot's default configuration in front of the table can be seen in Fig. 6.5. The latter three poses are derived from this default position. The exact definitions are set by empirical measurements. In the experiment the robot is given additional freedom to parametrize the pick-up action, namely the pre-condition of its pose. The robot executes segmentation and recognition, as well as grasping an object

on the table. The grasp-planning and perception results are measured and compared. They can be seen in Fig. 6.6.

Simulation here is performed offline. The results show significantly improved recognition results in the torso-down position and an increase in grasping quality of 32.5% in the torso-up position. The grasping quality indicator is measured in the standard ROS manipulation modules and expresses a relative metric of individual grasps. The recognition probability indicator is also measured in the standard recognition modules and expresses the likelihood that an object is similar to a 3D model in a CAD database. This experiment shows that it is possible to predict the results of standard robot actions and to reason using those predicted results.

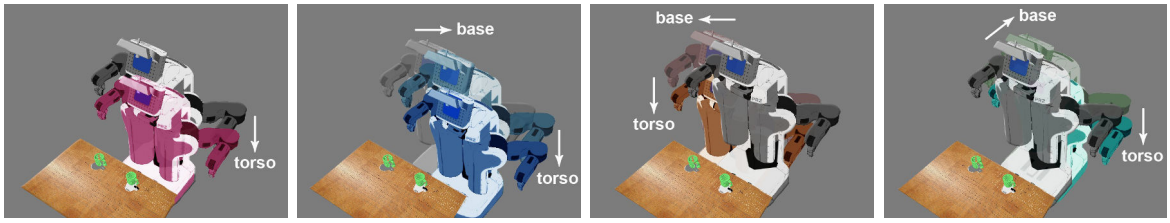


Fig. 6.5: Overlay of different poses tested within the simulation. The results of the simulation are used to aid object recognition and successful motion planning for grasping objects. This image shows four tested poses with the robot in front of a table on which there are some graspable and recognizable objects.

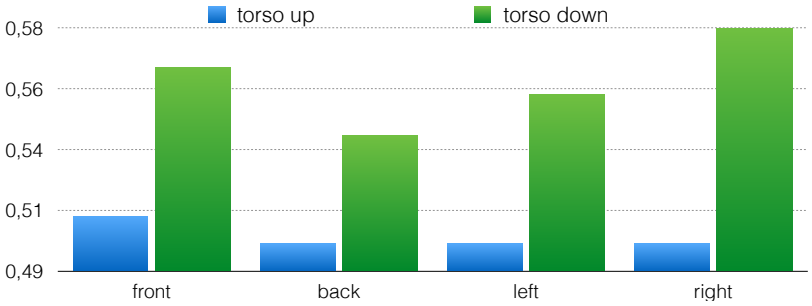


Fig. 6.6: Object recognition results (skill probability) for different robot torso and base positions: Objects can be recognized most reliably in the torso-down position. The base position in front of the table has little effect on recognition probability.

6.2.2 Table Occupation

In a restaurant, a typical task for the waiter might be to serve a coffee in front of a guest. The setup of the physical environment is illustrated in Fig. 6.7. The robot is given the instruction `achieve serve_coffee_to_guest_Task guest1` through its Command Line Interface (CLI). The planner is triggered and creates the plan shown in Listing 6.1 according to the problem description and current knowledge. Each line

represents a plan operator with its arguments. The plan operators correspond to robot capabilities and are bound to the robot. Arguments are instances from the robot's ontology and are stored in the central component of the RACE architecture, the Blackboard. The Blackboard collects and distributes information between different system components. Fig. 6.9 shows the robot's environment with respect to its knowledge.

```
1  !tuck_arms armtuckedposture armtuckedposture
2  !move_base premanipulationareaeastcounter1
3  !move_torso torsoupposture
4    !tuck_arms armtuckedposture armuntuckedposture
5    !move_arm_to_side rightarm1
6  !move_base_blind manipulationareaeastcounter1
7  !pick_up_object mug1 rightarm1
8  !move_base_blind premanipulationareaeastcounter1
9  !move_torso torsomiddleposture
10   !move_arms_to_carryposture
11 !move_base premanipulationareasouthtable1
12 !move_torso torsoupposture
13   !move_arm_to_side rightarm1
14 !move_base_blind manipulationareasouthtable1
15 !place_object mug1 rightarm1 placingareawestrighttable1
16 !move_base_blind premanipulationareasouthtable1
```

List. 6.1: Generated (parallel) plan for serving a coffee.

```
11 !move_base premanipulationareanorthtable1
12 !move_torso torsoupposture
13   !move_arm_to_side rightarm1
14 !move_base_blind manipulationareanorthtable1
15 !place_object mug1 rightarm1 placingareawestlefttable1
16 !move_base_blind premanipulationareanorthtable1
```

List. 6.2: Generated (partial) alternative plan for serving a coffee. The adapted operator parameters are highlighted.

When the robot approaches the table from the south, as illustrated in Fig. 6.7, plan step 15 instructs the robot to place the mug on the table. This fails because various objects occupy the placing area and this would cause the complete plan to fail. HIRES is designed to reason about the next best action and to prevent the plan from failing. In the scenario presented, the "place" action is considered crucial and is the most likely to fail. Therefore it is chosen to be reasoned about before it is executed in reality.

Upon successfully picking up the mug (plan step 7) HIRES is started. It uses the post-condition of the 14th plan step (the robot being at the manipulation area south of the table) and sets up the simulation accordingly (Fig. 6.8) in order to execute the next plan step (place the mug). This fails in the simulation too but this failure will not affect the original plan. Moreover, it provides the plan with failure information before actual failure. The failure is detected by the return value of the "place" action in the simulation. The planner, in return, is able to repair the current plan (a.k.a.

Fig. 6.7: The environment setup includes the counter, table and various table-top objects.

replanning) and to produce a (partial) new plan, as shown in Listing 6.2. This is possible because the two different manipulation areas (north and south) are valid parameters to the move base operator. As this work does not focus on particular replanning methods, for the sake of simplicity the planner creates partial plans before executing them. This allows replanning by simply changing the plan parameter and planning again (partially).

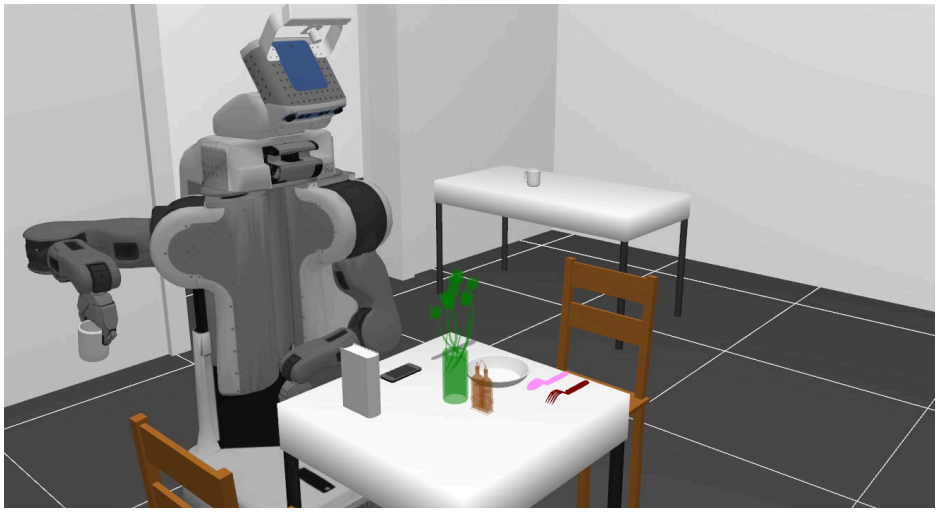


Fig. 6.8: The robot reaches the table to place the mug in front of the guest (omitted here, supposed to sit at the side of the table to the robot's left). As the placing area is occupied by various items the robot is unable to succeed.

As indicated in Table 6.3, plan execution fails while simulating serving to the PMA south of table1 (as it would in reality). Subsequent reasoning, which decides to serve to

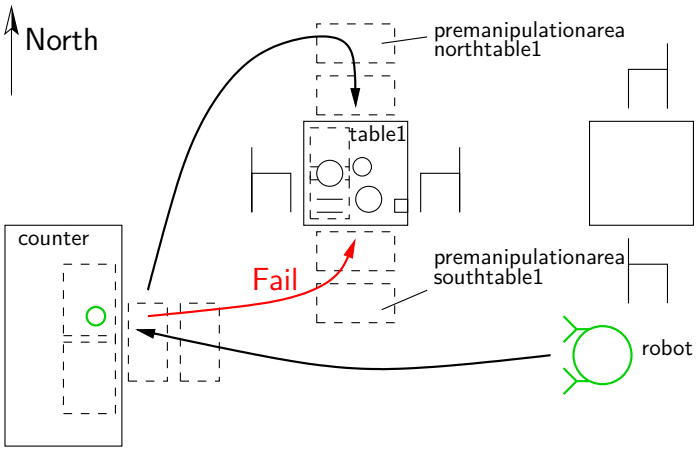


Fig. 6.9: Planning scene consisting of bounding boxes and coordinates for the table and counter and their approach positions. Labels correspond to plan operator parameters.

the other side of the table, succeeds. To evaluate the proposed system, the execution time when using imagination is compared in Table 6.4 to a standard system using replanning. The latter, illustrated in Fig. 6.10, shows that execution with imagination is faster than letting execution fail south of table1 followed by replanning. The detailed approach is described in Chapter 5.

move_base parameter	premanipulationarea *southtable1*	premanipulationarea *northtable1*
Plan	fail	ok

Table 6.3: Qualitative results: serving a coffee.

System	Default system + re-planning	HIRES
Duration	298 seconds	283 seconds

Table 6.4: Averaged quantitative results of five scenario runs.

The standard system does not include replanning at this time. For these experiments, replanning was simulated by issuing a new 'intend' causing a new partial plan to succeed. Notice that on average, imagination takes only 20 seconds compared to 90 seconds for replanning: ~80% faster.

In this simple example, only the simulation knows in advance that an object occupies the target table position (the object that leads to placement failure); the planner is unaware of the object. In this scenario the planning domain has been constrained to know only the initial position of the mug. The Planner knows nothing about objects

occupying the table. The simulation, in contrast, is able to update its internal world model with objects perceived from the robot's sensors[2] (cf. Section 5.6). In a realistic scenario this information must be obtained and fed into the simulation in order for it to be used efficiently. Nevertheless this example shows how a simulation-based robot action result can alter plan execution, reduce overall execution time and increase the probability of success.

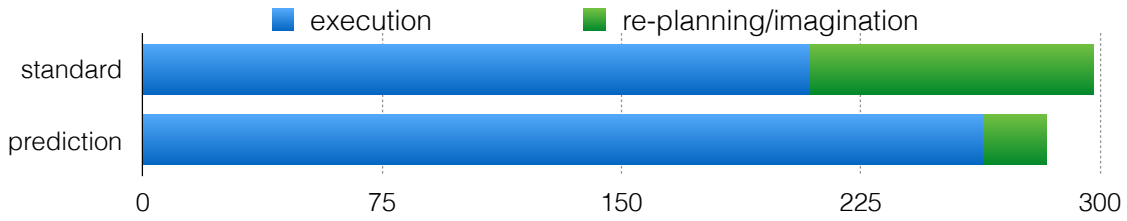


Fig. 6.10: Comparison of the prediction-enhanced system to the standard system with replanning, here the execution time is in seconds. Green indicates the time used for both replanning and imagination. In this scenario the prediction-enhanced system is about 15 seconds, or 5%, faster.

6.3 Physical Prediction and Probabilistic Method

The system proposed in this work is capable of different aspects of prediction. Physical prediction here refers to the prediction of action results that are neither temporal ("How long does a robot action last?") nor spatial ("Can the robot place an object there next to other objects on the table?"). The predictable actions referred to as "physical prediction" typically involve object dynamics only to enable a hypotheses about the action results.

6.3.1 Dynamics-based Prediction

For the robot to learn how to move smoothly, example-based compositional learning (EBCL) [NHRL14] was used to study robot dynamics data. For the transport of a tall object on a tray, the data turned out to be too noisy to show any significant co-relation between object or base dynamics and topple events [DDO+14]. Dynamics data from the experiments can be seen in appendix section A.2.

Some methods of filtering noisy data are briefly described below. Nevertheless, learning or predicting from robot dynamics is not the focus of this work.

[2] The author knows of no one-shot planning system that incorporates dynamic updates from the environment during execution and which updates the running plan, i.e. replanning.

[Gui08] presents a study of noise detection and filtering for borehole drilling data. Three types of filtering were compared: moving average[3], median and Butterworth. The results show that although most of the analyzed data, such as drilling speed, rotation speed and time data, contained a lot of noise, the noise could be removed using an appropriate digital filter. Most of the noise could be eliminated using time-domain filters, such as moving average and median filters. It was also shown that an auto-correlation analysis could determine the filter parameters.

6.3.2 Sampling-based Prediction

In this section, a system is evaluated that combines realistic physical simulation with symbolic reasoning. The main focus is on how to integrate *functional imagination* [MKNH08] into the HIRES framework [RKZZ14] with an off-the-shelf task planner and SEM to simulate an action before it is executed. The task planner has to create plans containing information about the actions to simulate. Furthermore, the plan needs to provide causal precondition-effect relations for actions. These relations will be used by a SEM to generate the context of the action relevant for imagination — the initial state imagined. From functional imagination, approximate information is obtained about the expected duration of an action and the likelihood of toppling. In order to use imagination, one must address the problem of how to express abstract qualitative terms (i.e. 'slow', 'topple') in sufficient detail for simulation, and how this simulation is performed. The SEM uses the expected duration obtained from imagination for temporal reasoning and scheduling (reasoning and scheduling are beyond the scope of this work).

A possible scenario is for the robot to fetch an object from a counter and put it on a tray in order to carry the object to its goal. Task planning and symbolic reasoning consider only an abstract description of the world state and the expected changes in this description resulting from executing a task, e.g., 'carry object'. The execution of the task is atomic - it may either succeed or fail. In contrast, the parametrization of physical aspects of the transition between abstract symbolic states is typically the domain of a physics simulation. By considering the trajectory, velocity and acceleration, the simulation can accurately model the carrying of an object in order to determine the parameters necessary to prevent toppling.

To evaluate the system, a variation of the latter scenario has been chosen: the robot is instructed to carry a pepper mill from a counter to a table in a restaurant domain. The scenario layout is shown in Fig. 5.16, in Chapter 5. In order to prevent the pepper mill from toppling, the correct parametrization (i.e. velocity and acceleration settings) for the `move_base` robot action must be 'imagined' in simulation. This action

[3] http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch15.pdf (March 16, 2015)

	θ_{fast}	θ_{slow}
all events	179054	171717
notopple	165906	165596
shaking	12735	6031
topple	413	90
duration $[s]$	18090 (∼5h)	17350 (∼4.9h)

Table 6.5: Experimental results: simulated toppling scenario measuring topple events for $fast$ and $slow$ parameter sets.

parametrization, in combination with the accurate time the action takes in simulation, can be used by the reasoning modules to take appropriate measures. One of the simulated action parametrizations might turn out to be superior to the parametrization chosen by the planner, e.g. it is more likely to succeed. In this case the execution will adapt the plan to use the safer parametrization (Fig. 6.11). Thus functional imagination is used to predict the outcome of future actions so that they can be adapted accordingly before execution.

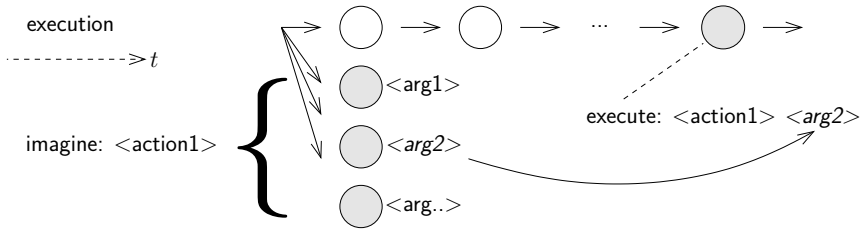


Fig. 6.11: Parameter instantiation in principle: during action execution (white circles) the result of one 'imagine' action (gray circles) might alter the parameters of another action, and so on. The 'imagine' action is executed in parallel with 'normal' robot actions.

Based on these definitions, the following validation experiments were executed in simulation: the robot was instructed to drive between fixed positions in a fixed order, repeatedly accelerating and stopping. The positions considered, which can be seen in Fig. 6.1, correspond to PMAs of furniture in the environment: tables (T1, T2) and a counter (C1). Multiple PMAs belonging to the same piece of furniture are distinguished by respective cardinal directions, e.g. ST1 for south and NT1 for north. The starting area of the robot nearStartArea1 (NSA1), was not included in the tests.

The robot's motion, its velocity and acceleration, as well as the tray object's deviation were continuously tracked. As expected, use of the *slow* motion parameter-set resulted in significantly fewer topple events, as shown in Table 6.5.

Based on these results it is now possible to calculate the likelihood of the three distinct events (topple, shaking, no topple) according to the equation: $\mathcal{L}(\theta|x) \approx P(X = x; \theta)$. Here x is defined by the three distinct values and θ is the (motion) parameter-set.

X	$\mathcal{L}(\theta_{fast}\|x)$	$\mathcal{L}(\theta_{slow}\|x)$	ratio (f/s)
notopple	0.9266	0.9644	≈1
shaking	0.0711	0.0351	≈2
topple	0.0023	0.0005	≈5

Table 6.6: Experimental results: simulated toppling likelihood $\mathcal{L}$ for $fast$ and *slow* parameter sets, as well as the relation between them: ratio (f/s).

The likelihoods are shown in Table 6.6 and are calculated as $\mathcal{L}(\theta|x) \approx x \cdot [\sum_{i=1}^{N} \theta_i]^{-1}$. Although the *topple* ratio between fast and slow results is ≈ 5 it is not guaranteed that such an event will happen in any time period as it has a probability by time of occurrence of once every $\approx 44s$. A *shaking* occurs every $\approx 1.4s$ with θ_{fast}.

Arbitration between the competing simulations is based on several factors. If the action returns a failure, the simulation is eliminated; if it is successful, the confidence measure is evaluated (Eq. 6.1) and higher confidence wins. Finally the duration measure is taken into account to favour faster execution. c, the result of the probabilistic model, represents confidence that the robot task will succeed. It is based on possible random events and some empirical coefficients.

$$\begin{aligned} c &= \left(\frac{a_t \cdot c_t + a_s \cdot c_s + a_n \cdot c_n}{a_t + a_s + a_n} \right)^{-1} \\ c = (0,1] &:= \{c \in \mathbb{R}^+ | 0 < c \leq 1\}, a_t + a_s + a_n > 0 \end{aligned} \tag{6.1}$$

Coefficients are: c_t (topple), c_s (shaking), c_n (no event) and the variables are: a_t-topple, a_s-shaking, a_n-no event. To keep the confidence between 0 and 1 and to scale well between the three possible events, the following coefficient values were defined empirically from the data of Table 6.5 and Table 6.6: $c_t = 100$, $c_s = 10$, $c_n = 1$. The duration of an action is simply defined by: $t_{action} = t_{end} - t_{start}$ with $t_{action\_max} := 300s$.

The Simulation Client monitors toppling events by sampling the object's deviation angle at a rate of approximately 20 Hz. Depending on the measured angle, the counter for one of the three possible events is increased. If an action fails, its corresponding confidence value will be set to 0. If no action (parametrization) has been successful, or if imagination has not concluded either before the defined deadline or before the SEM has to execute the (parametrized) action, a standard value, i.e. $fast$, is inserted to instantiate the complete plan.

6.3.3 Results

The results from executing the proposed scenario on a PR2 are presented in the following section. The time resources used by the simulation during real scenario execution

are listed in Table 6.7. Fig. 6.13 shows the time taken by each involved action and the cumulative time. The result of the scenario is that the robot employed a 'slow' movement when carrying the tall object on the tray. This recommendation was provided as the optimal motion parameter set by functional imagination. An average confidence result is $c_{fast} = 0.87$ including 12 shaking and $c_{slow} = 1.00$ with only 'notopple' events. The scenario setup is shown in Fig. 6.12.

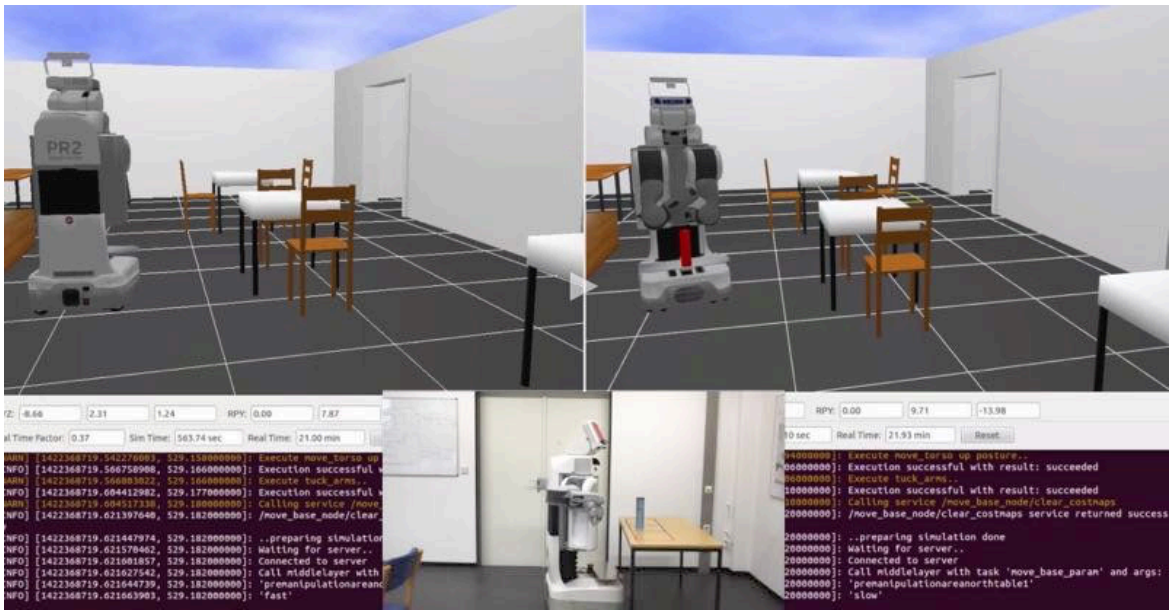


Fig. 6.12: Real toppling demo: while the robot is executing the pick-up (bottom) two simulations (top) are testing a *fast* and a *slow* parameter set.

The two 'imagine' tasks are performed in parallel (both at the same time). They occur after the first "prepare move base" (actually they have already been triggered during that action) because the imagine tasks can theoretically run for an arbitrary period (in practise the duration is more limited as only short actions are simulated, not complete plans). The 'imagine' tasks should therefore be started as early as possible during execution. All information for the imagine action is available at the start of plan execution: the projected world state can be created with complete information about the pepper mill - what and where it is and where to take it. In other scenarios this information might only become available later, and so imagination might be started later during plan execution.

The robot has to leave the counter or table before being able to place an object onto its tray because it stands too close to the table to move the arms and not collide with the counter or table. Similarly, the robot must lift the object from its tray before approaching the table. The introduction of an artificial PMA to represent this constraint in the planning domain was described at the beginning of this chapter and is shown in Fig. 6.1.

Notably the figure shows that both imagination actions return before the robot picks up the object from the counter. Therefore information gained from imagination can be used to schedule and reschedule any action after picking up the object.

	θ_{fast}	θ_{slow}
Duration simulation [s]	32.01	45.27
Duration real [s]	76.24	109.85
Real-time factor	0.42	0.41
Duration scenario [s]	384.0	
Relative imagination time	0.08	0.12

Table 6.7: Experimental results: time resources used by simulation during the toppling scenario.

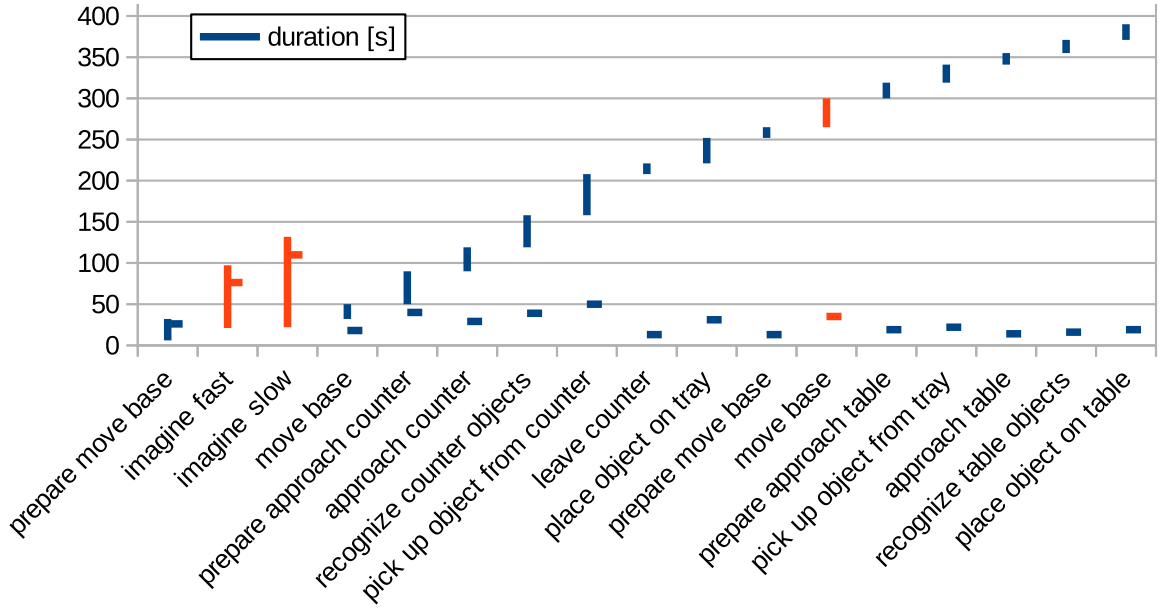


Fig. 6.13: Topple scenario: robot actions over time in seconds. The two imagination actions, executed in parallel, and the affected move base action are highlighted in red.

6.4 Summary

This chapter has shown that Gazebo simulation is deterministic in the sense that multiple simulation runs give consistent results. Nevertheless it has also shown that the influence of OS process scheduling can lead to slight variations in simulation results. Related work on verifying simulations suggests that simulated results can be compared with real-world results. The experiments presented here have shown that

simulation results from the selected physics engine (ODE) can accurately predict real-world states. Related work suggests that ODE performs well in comparison with other physics engines.

These results have shown that offline simulation can be of benefit in obtaining optimal parameters. The results from various experimental scenarios show merit in the causal, temporal and spatial aspects of simulation. The different aspects of the proposed prediction system have shown that prediction aided by simulation can prevent plan failures, i.e. by plan change or replanning. It has also been shown that the resulting system is more efficient, i.e. actions execute in less time, supporting the hypothesis stated in Section 1.4. The final chapter, Chapter 7, presents some conclusions and an outlook.

Conclusion and Outlook

7

This chapter summarizes the evaluation and findings of this work. It discusses the value of the results and finally looks at future developments.

This work set out to answer the research question stated in Chapter 1: *How can a task planning based robot system be improved by prediction derived from physical simulation?* A precondition for success was that simulation is accurate enough to allow comparison with real world execution. The very similar test durations in simulation and on the real robot (shown by the results in Section 6.1) indicate that such comparison is indeed valid and support the use of the selected physics engine.

To answer the research question, the method of "functional imagination" was integrated: a realistic prediction of robot activities, based on physical simulation, was integrated into a plan and execution-based system (Chapter 5 describes how it is done, i.e. the approach, cf. Section 1.4). As the results in Chapter 6 show, this system adapts its plan (or replans) before executing plan actions that simulation shows are likely to fail.

The PR2 mobile robot platform is accurate, extensible and well suited as an evaluation platform. Hardware extensions are straightforward to apply. This allowed the mounting of a tray (to carry additional objects) and of extra sensors (such as a laser range finder and a thermal camera) as well as the addition of auxiliary robot capabilities that allowed objects to be carried with the robot arms while driving. The robot was able to execute all scenarios related to the evaluation of this work.

As a basis for the software framework used in this work, a distributed ROS-based robot control architecture was developed in the context of the RACE project. This was extended to incorporate a physics-based simulation that predicts robot action results with a high-level, plan-based, cognitive system.

To compensate for execution speed limitations caused by simulation, a method that interleaves simulation and real-world action execution was chosen. A probabilistic approach was used to project perfect simulation results onto the real world. The state of each simulation was sampled during execution to obtain the probability distribution of possible outcomes. The simulation granularity, in terms of action parametrization, remains limited by execution speed to certain planned actions and alternatives.

The hypothesis stated in this work was:

A system integrating such prediction is more efficient (such as requiring less time, plan steps) than a comparable system without.

The experiments in Chapter 6 showed that various failure conditions could be detected and that such failure prediction could lead to improved overall system robustness (the scenario could be completed) and execution time. Similarly, in physical prediction scenarios the use of simulation allowed the robot to carry objects without the object toppling. This improves the robustness of robot actions and allows tasks to be carried out more efficiently. The results verify the hypothesis.

In principle, the computational cost of simulation limits the sample size and the duration of actions. Nevertheless the practical scenario presented showed that interleaving simulation with execution does not necessarily incur extra costs.

7.1 Conclusion

The main experimental findings are summarized in Chapter 6. This section assembles the findings to answer the study's research question: *How can a task planning based robot system be improved by prediction derived from physical simulation?*

1. **Functional imagination methodology:** The system architecture presented here combines the traditional SPA control paradigm with functional imagination to create a high-level, plan-based, reasoning and execution system (cf. Section 5.1). This has been demonstrated in a conceptional and experimental way.
2. **Plan operators:** The limitation of simulating only plan actions instead of (partial) plans appeared to be purely qualitative, as not all actions must be simulated. Assigning each imagination action to an imagination operator in a symbolic HTN planner provided an abstract and adaptable method (cf. Section 5.2).
3. **Handling uncertainty:** The proposed probabilistic approach samples simulation to accommodate uncertainty and allows uncertainty to be represented as a confidence value (cf. Section 6.3).
4. **Simulation accuracy:** The experiments verified that in simulation, robot actions execute accurately enough to allow the comparison of simulation results with reality (cf. Section 6.1).
5. **Distributed system approach:** The distributed system architecture presented here provides the required performance and compensates for the resource bottleneck imposed by sophisticated physical simulation (cf. Section 5.2). The performance scales linearly with the number of actions to be simulated $\mathcal{O}(n)$, and is constant when actions are executed in parallel $\mathcal{O}(1)$, in the sense that additional hardware can be added on demand.

6. **Common Sense:** Experiments testing recognition and manipulation of the environment using different robot postures demonstrated the potential for physical simulation to enhance competence (cf. Section 6.2.1).
7. **Action parametrization:** The parametrization approach showed that action failure (and thus possible plan failure) could be prevented and that the plan could be successfully completed. Execution measurements demonstrated that although simulation is expensive, overall system execution time could be improved (cf. Sections 6.2.2, 6.3.2).
8. **Training data generation for learning:** The proposed automated execution framework ASEF supports the creation of training data from robot experiences (cf. Section 5.7).

While compliant with state-of-the-art approaches (cf. [KDB11, dSPA13]), this work differs significantly in the way simulation is applied to the finding of optimal action parameters and to the discovery of a suitable robot plan. Its main contribution is the use of simulation as a general companion tool to robot reasoning. Moreover, it uses simulation during (plan) execution in order to adapt current behavior to unforeseen events in the robot environment. And it contributes to the understanding of the probabilistic nature of real world uncertainty in robot action results.

Evidence from this work and that of several others, including Beetz [KHB12, KDB11, KDGB11, Bee00, MB09], indicates that simulation is a suitable companion for improving real world robot competence, beside being a helpful tool in developing and evaluating software for robots. Empirical findings in this work show that simulation can be used to represent the world state, to reason about it and to adapt robot behavior on-the-fly. The theoretical arguments in favour of this method could influence future methodologies in the design of robot control programs and could thus support cognitive improvement of future systems.

7.2 Limitations and Future Research

The proposed simulation-based prediction approach is capable of testing all robot actions before they are executed in reality. The current approach is limited to 'imagining' one action a time, which makes functional imagination strongly dependent on plan-step granularity; granularity is based on the underlying atomic robot capabilities, such as moving the robot to a destination. A valid future research question, therefore, is how functional imagination could benefit from either intra-plan-step prediction or partial-plan prediction. An example could be whether parametrization of the robot's drive action can be extended to adapt to a global optimized trajectory. This can be realized by dividing a trajectory into multiple parts, for each of which a simulation provides the optimal motion parametrization. This approach should lead to faster

overall trajectories, as only part of each trajectory might have to be executed with a slow-motion parameter set.

The current approach uses simulated action results in order to parametrize actions of an existing plan. If an action fails in all tested configurations, it will be necessary to produce a new plan, i.e. to perform re-scheduling. Therefore future research might investigate whether a re-scheduling integration approach improves robot competence.

Action durations are determined during simulation (temporal prediction), but the present system takes no advantage of these (although they are quite accurate, cf. Section 6.1). For example, in the case of temporal-constraint execution monitoring, an action must be executed before or after another action, or it might not be executed at all. Thus taking simulated action durations into account during execution also enforces re-scheduling.

Other future research questions arise from the development of faster computer hardware and more complex scenarios: How can parallelization help a plan benefit from faster distributed systems? And how does the use of greater computational power affect action simulation granularity? In a recent article [Pra15] the author sees a "Cambrian Explosion-like" development in robotics, partly coming from cloud-computing. "Learning from Imagination" is supposed to be one of the biggest ideas exploiting cloud-computing for robotics. Computational simulation power helps therein

> ... to explore circumstances that may be faced by a robot in the future and to experiment with possible solutions, remembering only those that worked. Such simulations can be done without the need for any physical activity, and every robot's dreams will improve the performance of all robots.

As a final future research recommendation, there is the question of how other physical simulation engines perform compared to the ODE engine used here. Qualitative performance and feature analyses might be helpful. For example other engines might support non-rigid-body interactions.

This thesis has presented an evaluative perspective of simulation accuracy and of the improvements achieved in mobile robot performance and competence. The inclusion of many more test samples would certainly strengthen the findings.

Furthermore this thesis has presented an approach by which common-sense physics knowledge can be exploited to prevent robot action failure in the field of mobile service robotics. It is hoped that this approach will help advance research and development and that it will support the creation of new service robotics applications. The work covered four major aspects in this context. Firstly, it defined and implemented a probabilistic method for the use of simulation as a direct model representing the robot's world state. Secondly, it validated the simulation of real-robot actions as an appropriate tool. Thirdly, it developed the concept of "functional imagination" using

simulation to predict robot action results. Finally, it designed an architecture that integrates "functional imagination" into a high-level, cognitive, plan-based system.

The system presented integrated high-level symbolic reasoning and concrete physics-based reasoning and ran in parallel on a PR2 robot and in the Gazebo simulation. The scenarios presented in Chapter 6 showed how physics-based common-sense knowledge can improve a plan-based high-level cognitive system via action parametrization.

In summary, this work proposed a methodology for the use of a physical simulation to make planning and reasoning systems more robust when dealing with real-world scenarios. Similar systems have in recent years made significant progress, incorporating dedicated geometric robot capabilities into the planning and reasoning process. This has already led to a significant increase in robot system performance. The innovation rate and the number of publications in this field indicate the potential importance of employing non-abstract reasoning in symbolic-based AI systems. This work provides a general approach by which systems can benefit from low-level, physics-based reasoning. This, it is hoped, will contribute to and accelerate the acceptance of simulation-based prediction in future intelligent mobile service robotics.

Appendix

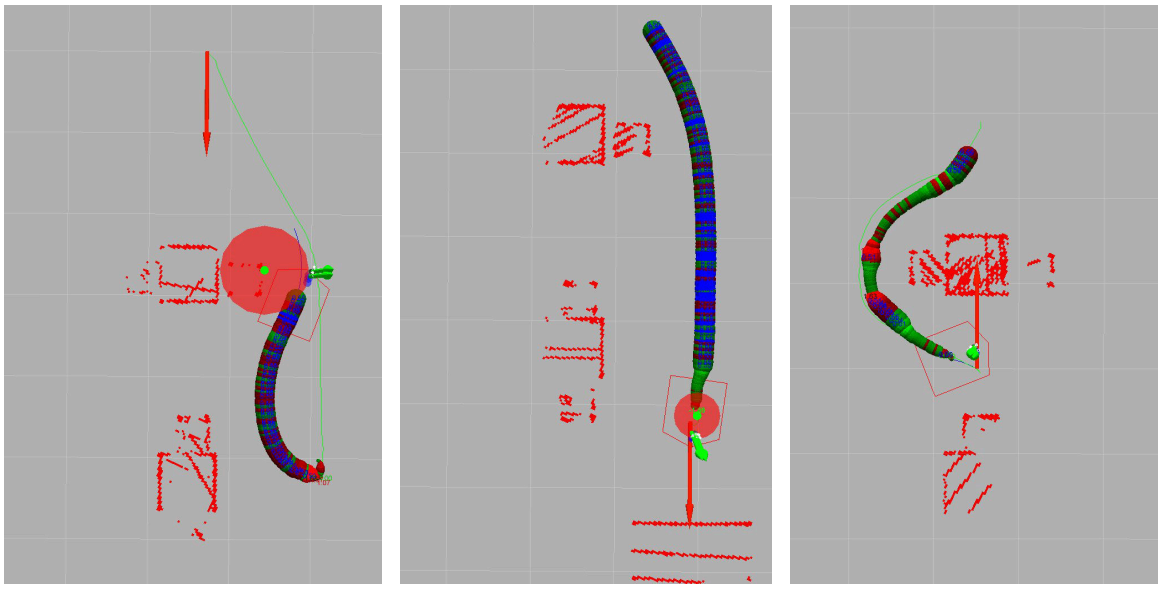
A

This chapter lists additional material related to this work.

A.1 Gazebo Simulation

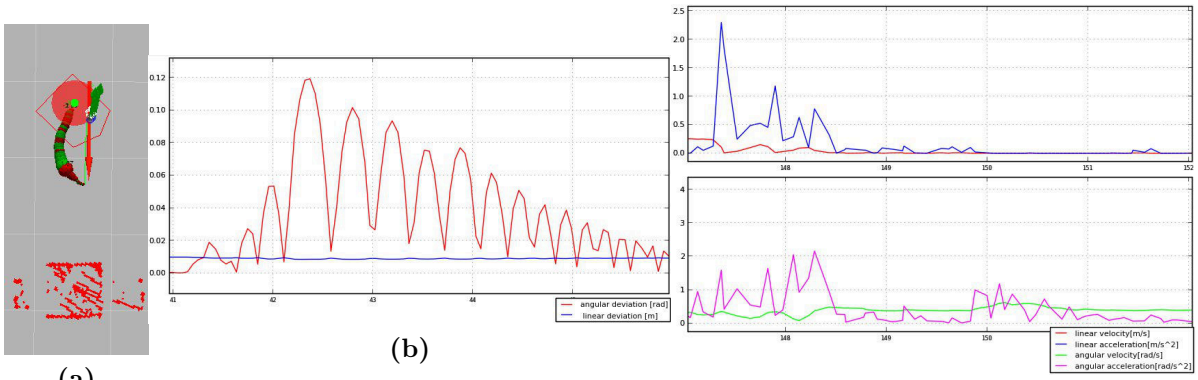
Fig. A.1: Restaurant simulation environment (top view). The robot, on the right side, is facing the entrance.

A.2 Dynamics Data



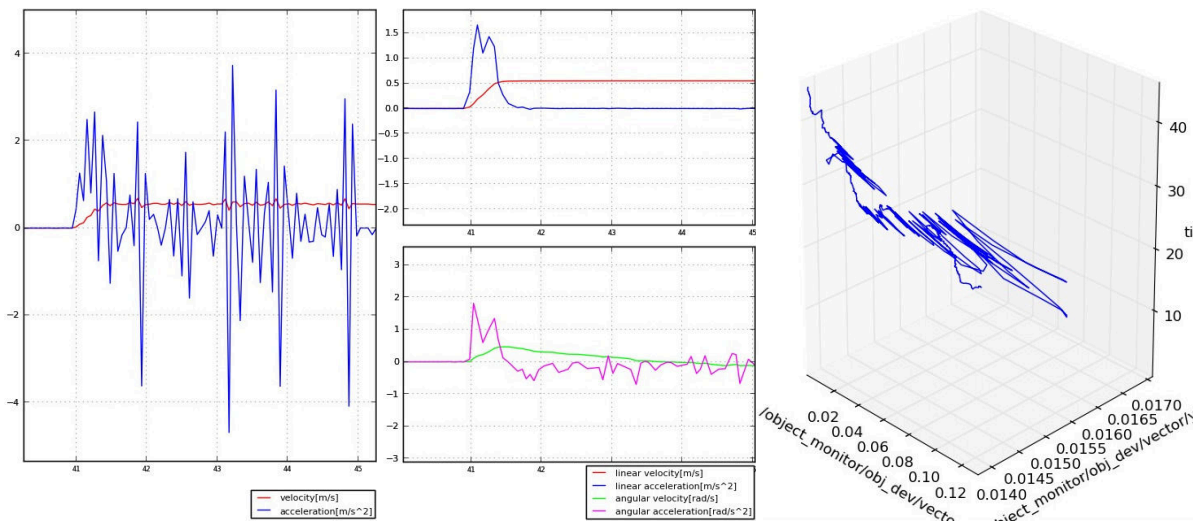
(a) (b) (c)

Fig. A.2: PR2 base dynamics data for various movements; the rotation center of the tray object is also shown.



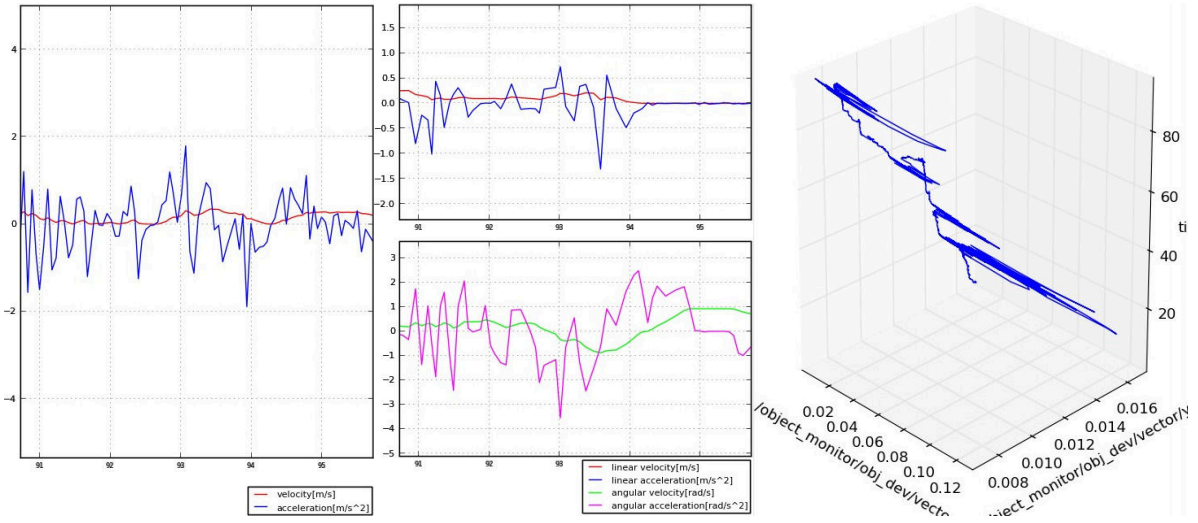
(a) (b) (c)

Fig. A.3: Shaking Event: Jerky motion (a) causes a tall tray object to shake but not to topple (b). The corresponding linear and angular robot base dynamics are shown in (c).



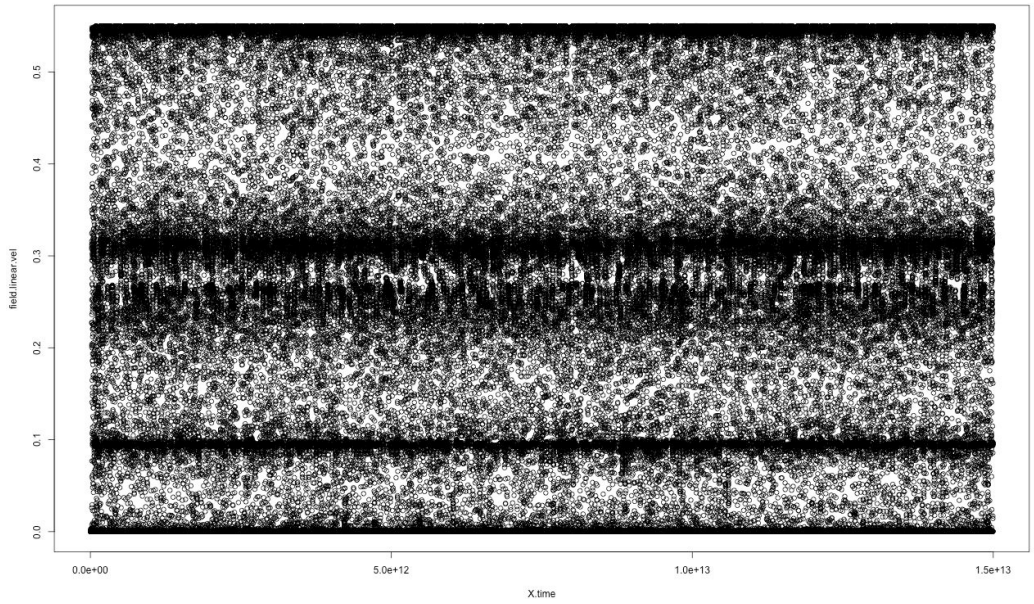
(a) **(b)** **(c)**

Fig. A.4: Dynamics of object on tray (a), robot base dynamics (b) and object deviation (angular and linear) from ideal orientation and position (c). See Section 5.8.1 for details on toppling detection.

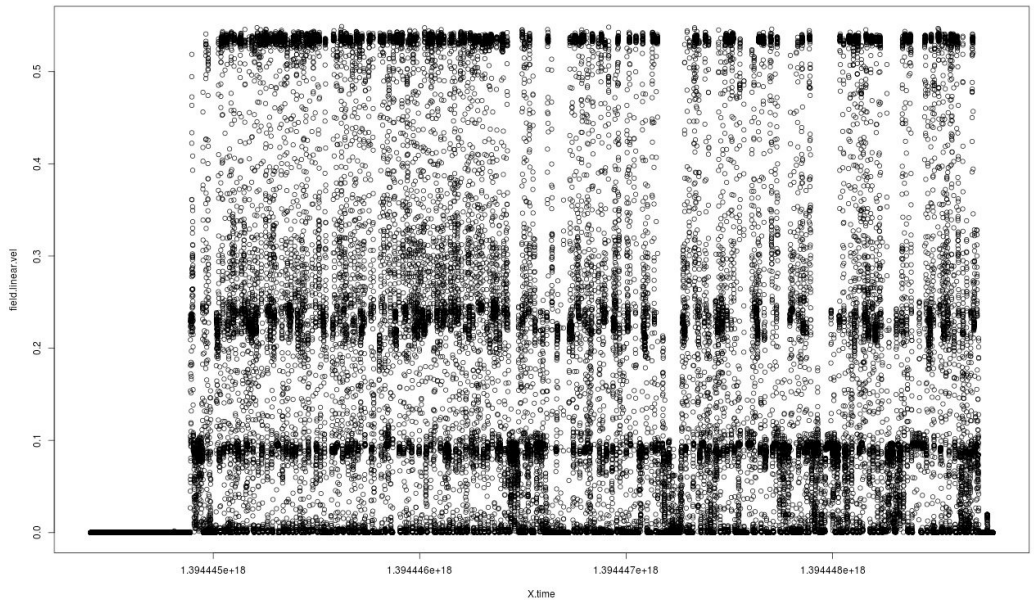


(a) **(b)** **(c)**

Fig. A.5: Dynamics of object on tray (a), robot base dynamics (b) and object deviation (angular and linear) from ideal orientation and position (c). See Section 5.8.1 for details on toppling detection.



(a)



(b)

Fig. A.6: Robot velocity distribution over time: a) Simulation, b) real. The velocity data was collected (in simulation) over 4 hours of random driving, equivalent to 1 hour in reality.

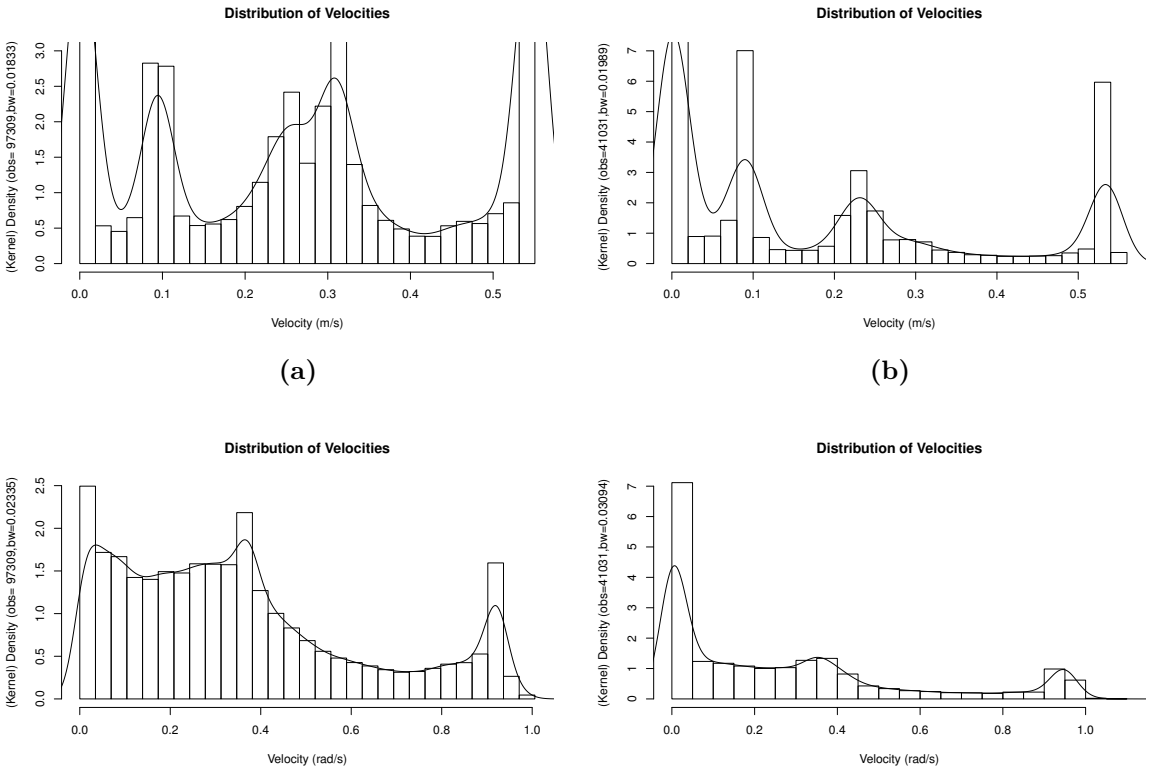


Fig. A.7: Linear and angular velocity in simulation and reality with a kernel density function: a) linear velocity in simulation, b) linear velocity in reality, c) angular velocity in simulation, d) angular velocity in reality.

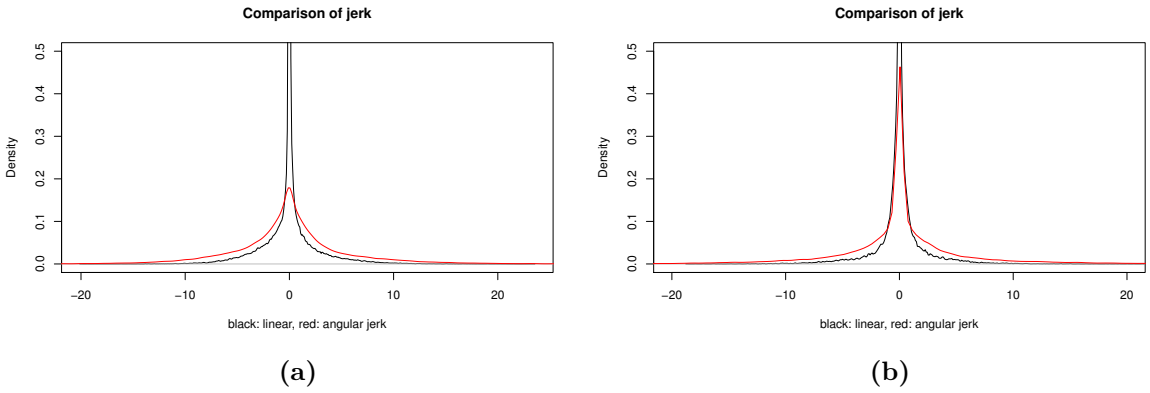


Fig. A.8: Robot jerk in simulation and reality: a) jerk in simulation, b) jerk in reality.

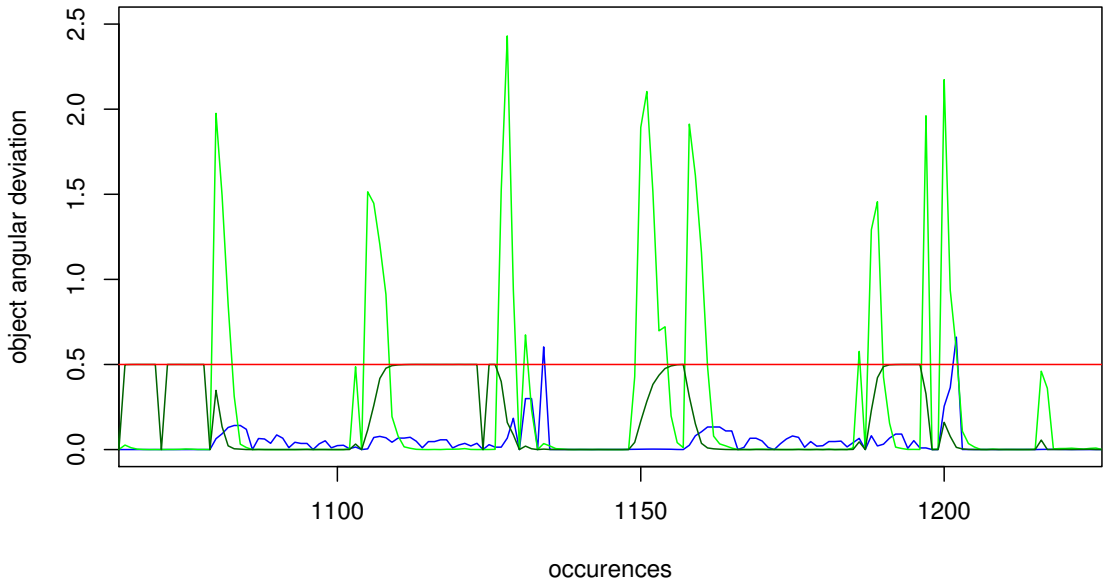


Fig. A.9: Object deviation vs. linear velocity/acceleration

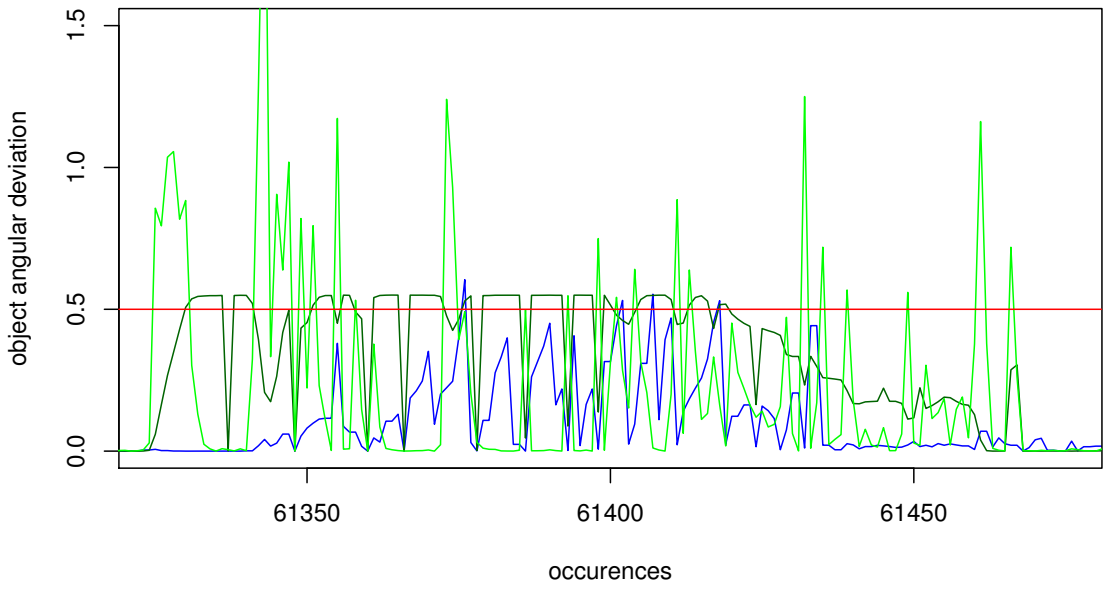


Fig. A.10: Object deviation with limited acceleration

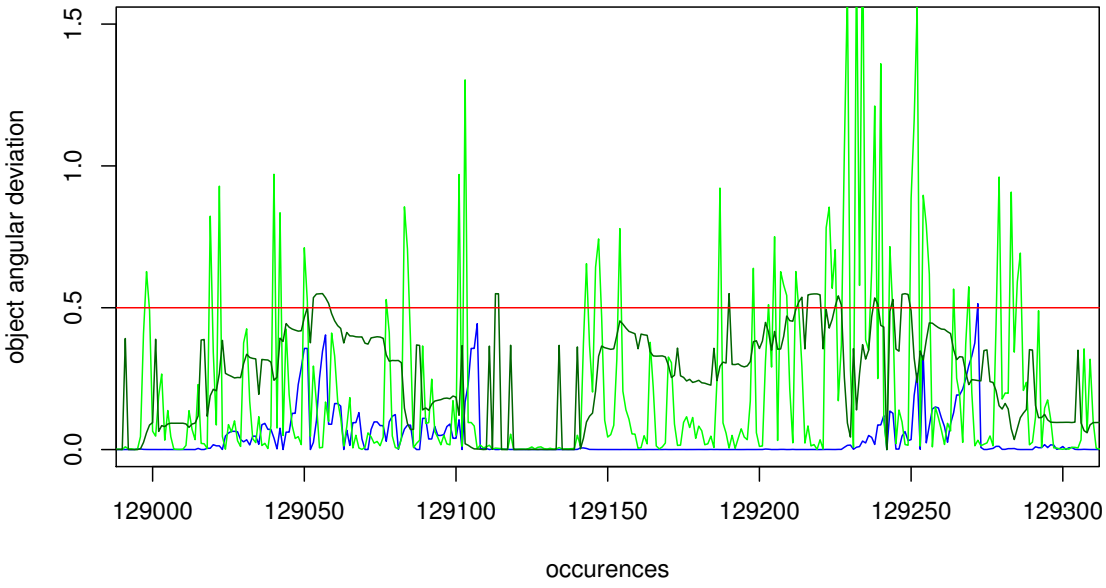


Fig. A.11: Object deviation with limited acceleration (complex)

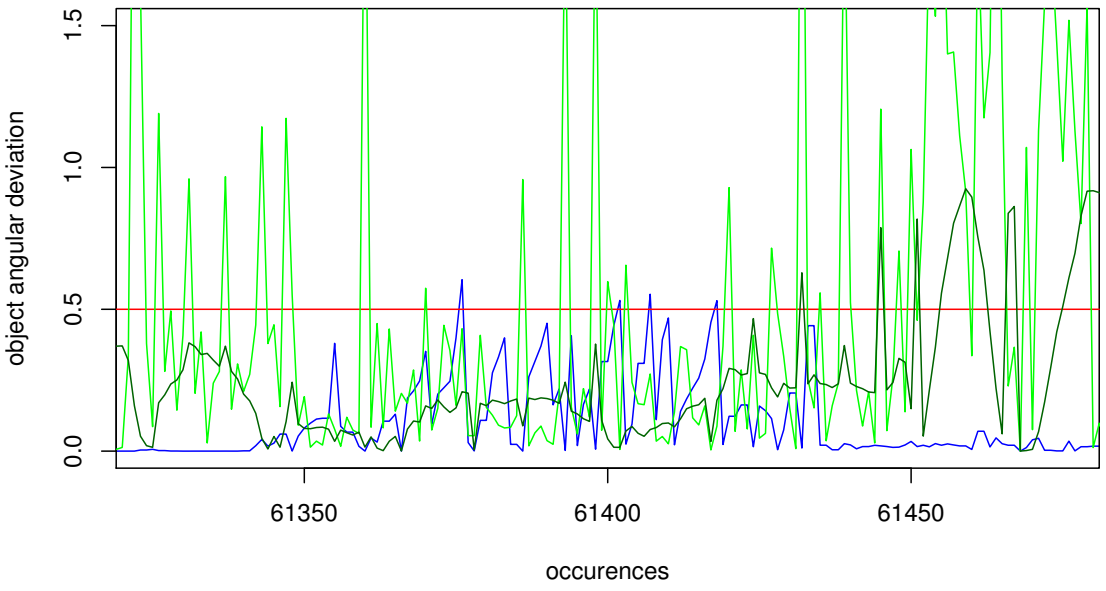


Fig. A.12: Object deviation with angular dynamics and limited acceleration

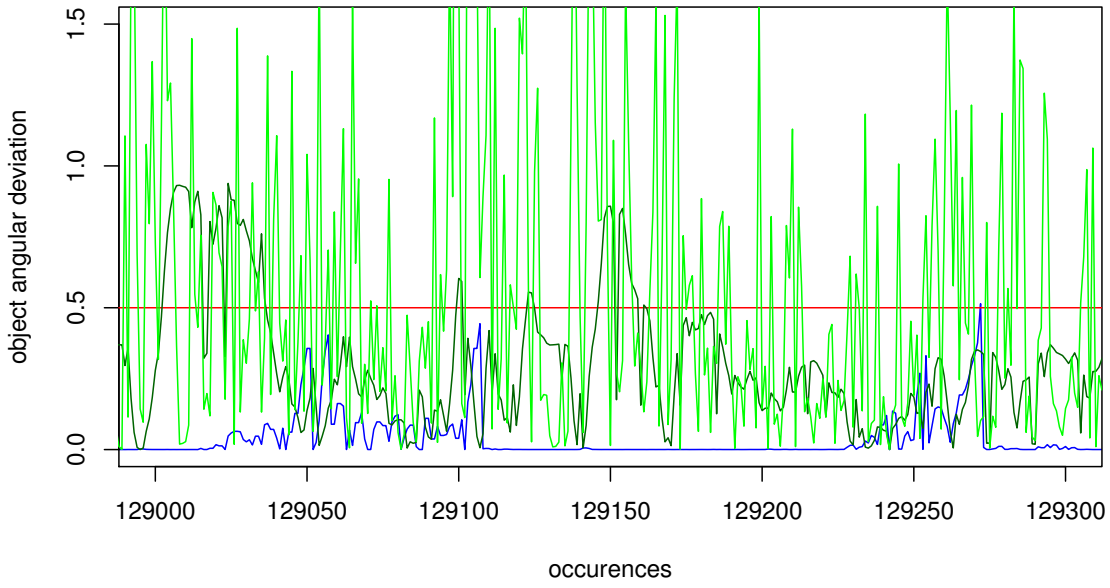


Fig. A.13: Object deviation with angular dynamics and limited acceleration 2

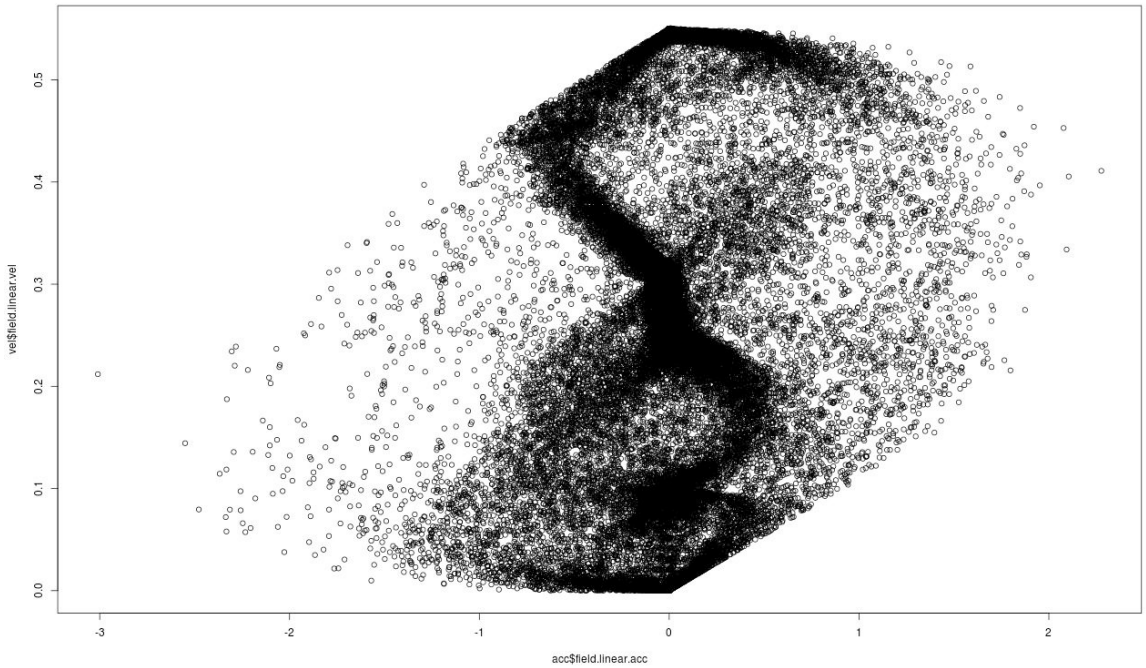


Fig. A.14: Relation between velocity and acceleration.

A.3 Creating a 2D Occupancy Grid Map with the PR2

```
# start PR2 or gazebo
# then:
roslaunch pr2_2dnav_slam pr2_2dnav.launch
# start rviz
roslaunch pr2_navigation_slam rviz_move_base_slam.launch
# save map:
rosrun map_server map_saver -f /tmp/my_map
```

List. A.1: Launching the 2D SLAM package for the PR2.

A.4 Moment of Inertia

```
<gazebo version="1.0">
<model name="race_peppermill" static="false">
  <link name="body">
    <origin pose="0 0 0 0 0 0"/>
    <inertial mass="0.5">
      <origin pose="0 0 0.15 0 0 0"/>
      <inertia ixx="0.003903125" ixy="0" ixz="0" iyy="0.003903125" iyz="0" izz="
 0.00030625"/>
    </inertial>
    <collision name="geom">
      <origin pose="0 0 0.15 0 0 0"/>
      <geometry>
        <cylinder radius="0.035" length="0.30"/>
      </geometry>
    </collision>
    <visual name="visual1">
      <origin pose="0 0 0.15 0 0 0"/>
      <geometry>
        <cylinder radius="0.035" length="0.30"/>
      </geometry>
      <material script="Gazebo/Red"/>
    </visual>
  </link>
</model>
</gazebo>
```

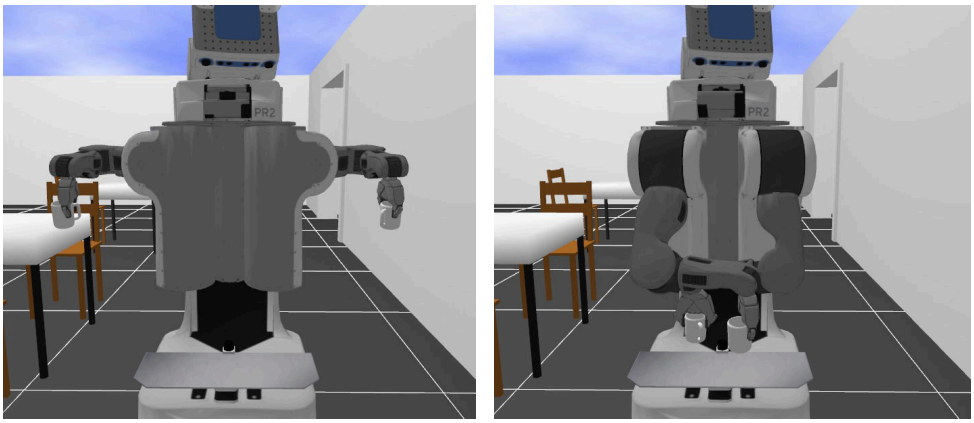
List. A.2: Pepper mill model with inertia parameters.

A.5 Standard Deviation and Normal Distribution

Here the random variable is X and the mean value is μ. The standard deviation is calculated as follows $\sigma = \sqrt{(X-\mu)^2}$. For discrete distributions the following equation is appropriate: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$, where $\mu = \frac{1}{N}\sum_{i=1}^{N}x_i$, for the same probability and for all x. The normal probability density function is as follows: $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$

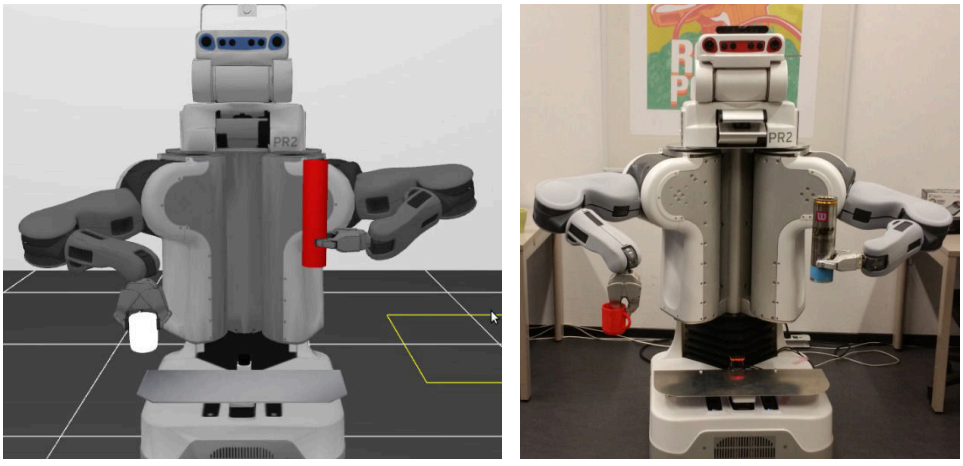
A.6 Additional Robot Capabilities

This section presents robot capabilities developed for the experimental scenarios in the contexts of the RACE project and this work.



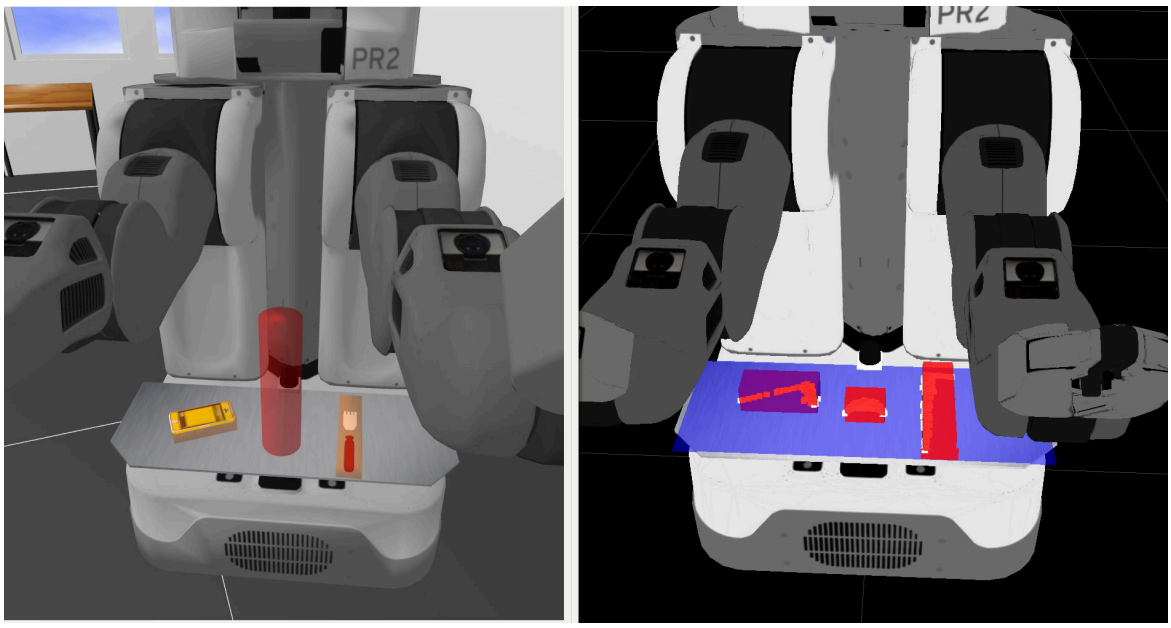
(a) (b)

Fig. A.15: Carry arm posture: the robot can grasp (a) two objects in its grippers and can carry the objects while moving. In this configuration the arms are within the robot's footprint in order not to collide with the environment. The arms are held out of view of the sensors so as not to be wrongly detected as obstacles (b).

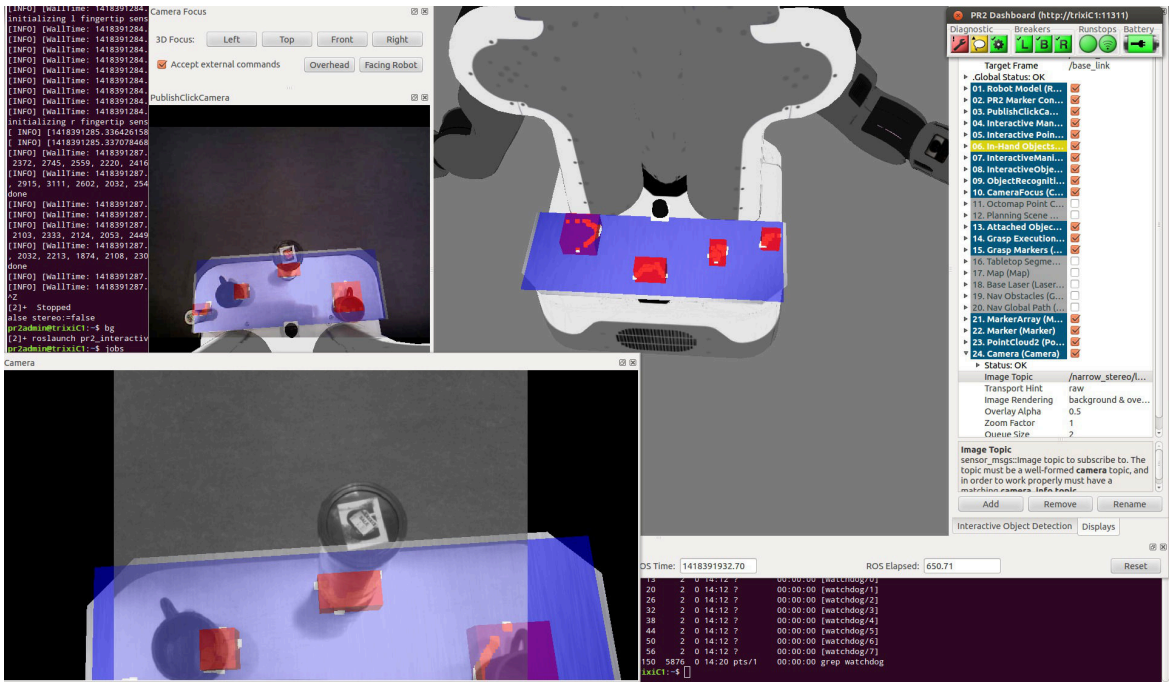


(a) (b)

Fig. A.16: Tray object manipulation: the robot can place objects on the tray and can grasp objects from the tray, in simulation (a) and on the real robot (b). The objects are detected by a laser scanner mounted at the level of the tray.



(a)



(b)

Fig. A.17: Tray object detection: objects are clustered and represented by bounding boxes (red) based on the laser readings in simulation (a) and on the real robot (b).

A.7 The Pepper Mill


Fig. A.18: A typical tall pepper mill found in a supermarket.

A.8 Statistical Analysis of Simulated Robot Actions

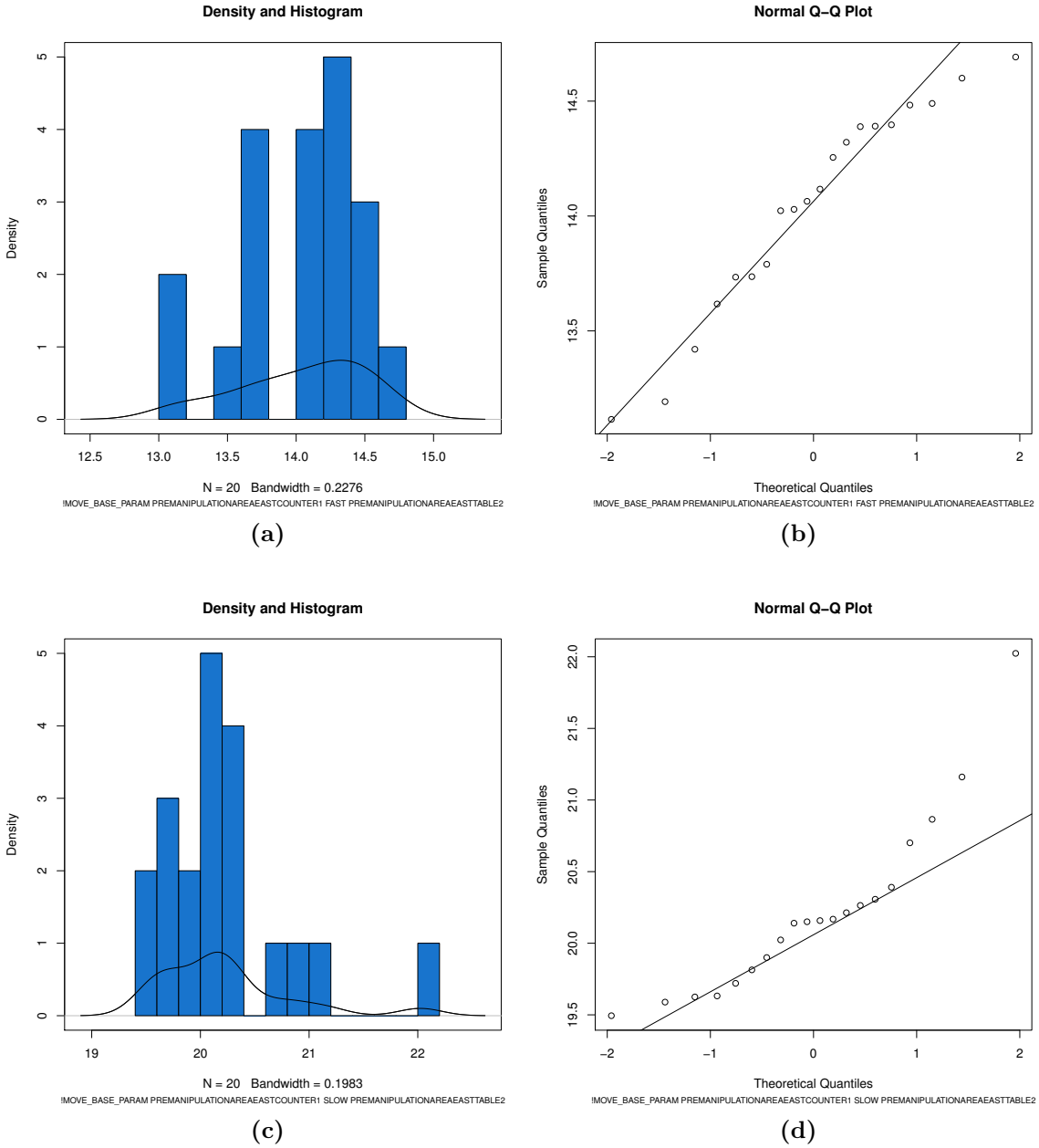


Fig. A.19: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

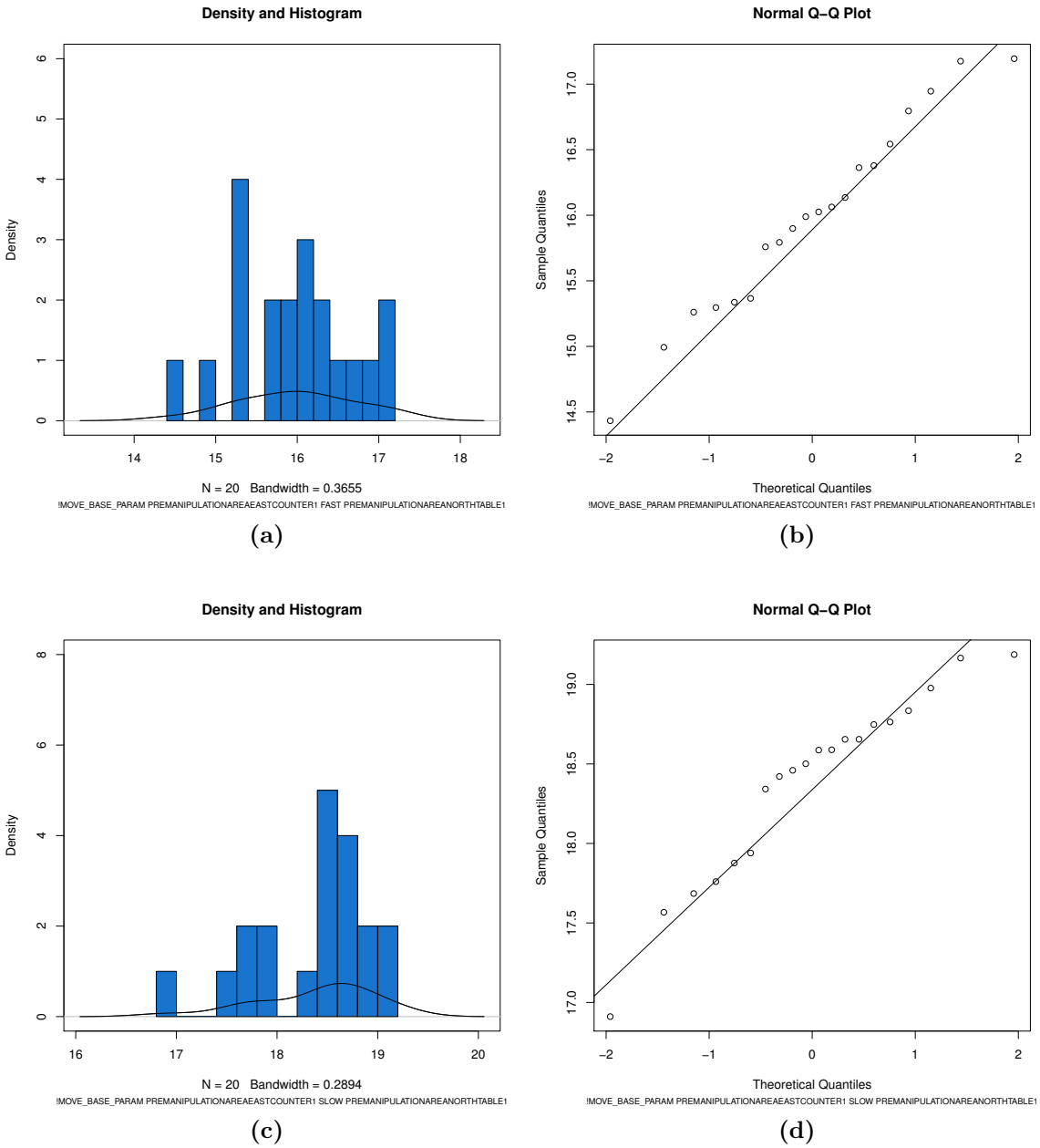


Fig. A.20: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

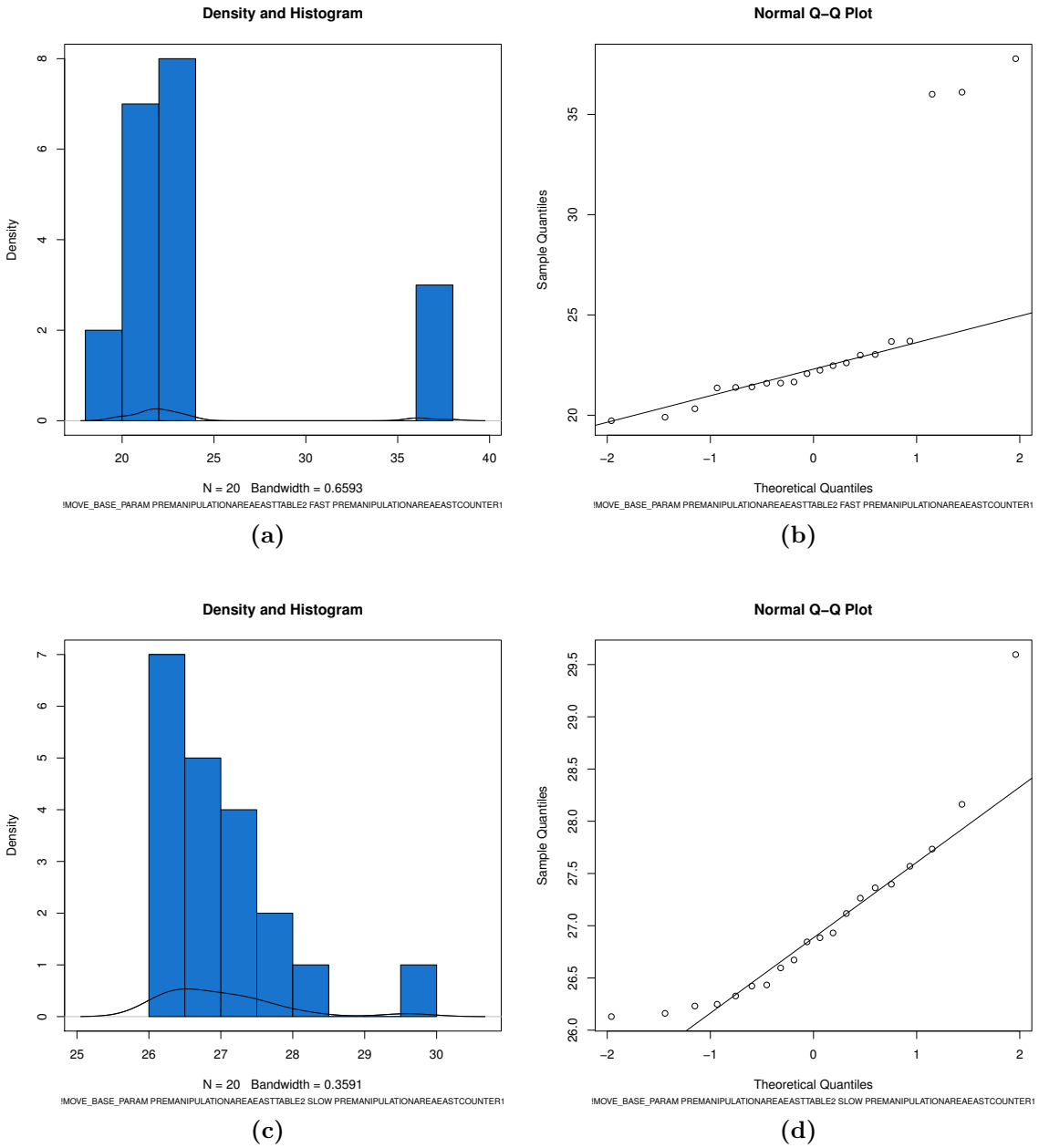


Fig. A.21: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

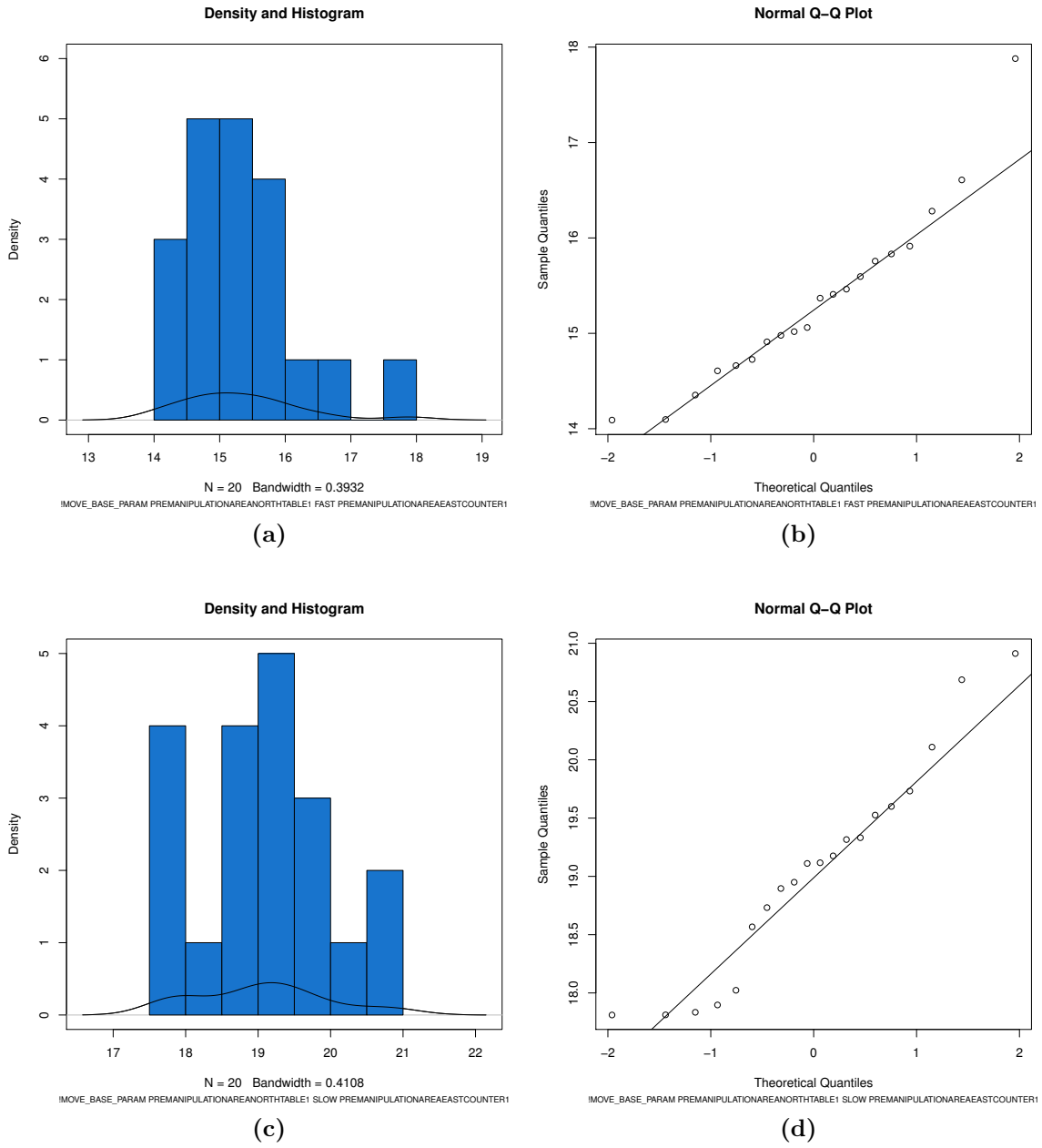


Fig. A.22: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

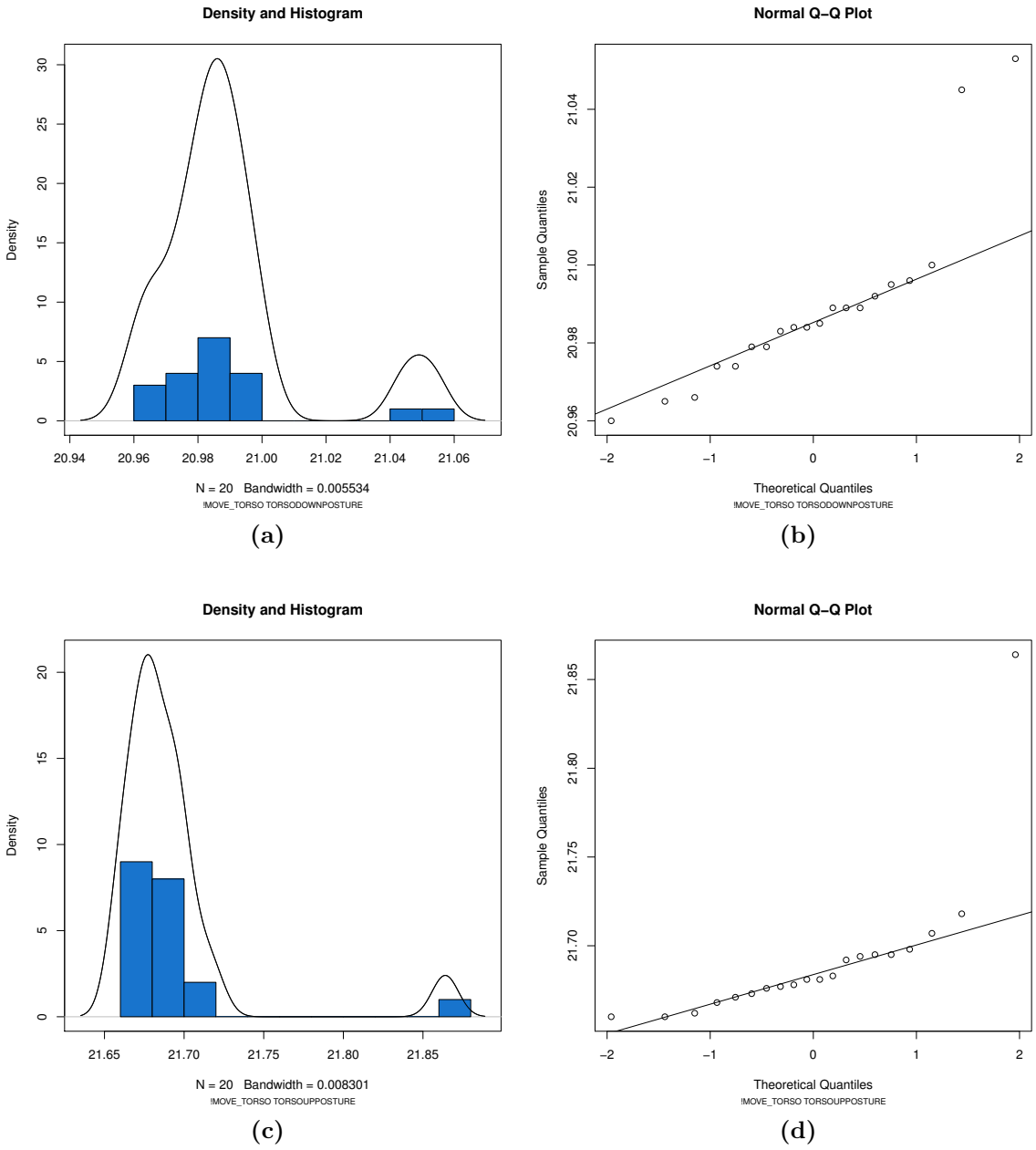


Fig. A.23: Statistics of action (a) Density and Histogram. (b) Q-Q Plot.

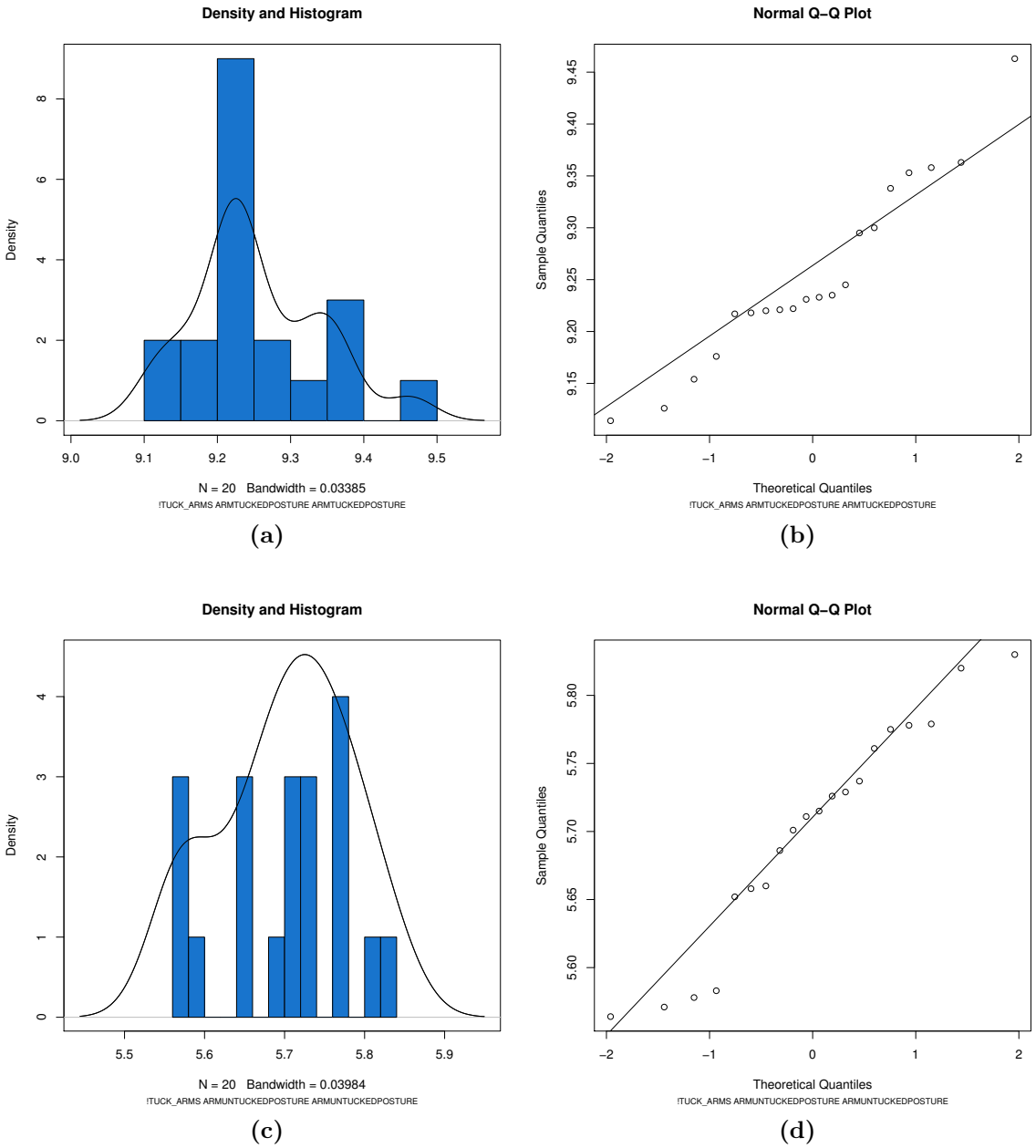


Fig. A.24: Statistics of action (a) Density and Histogram. (b) Q-Q Plot.

A.9 Statistical Analysis of Real Robot Actions

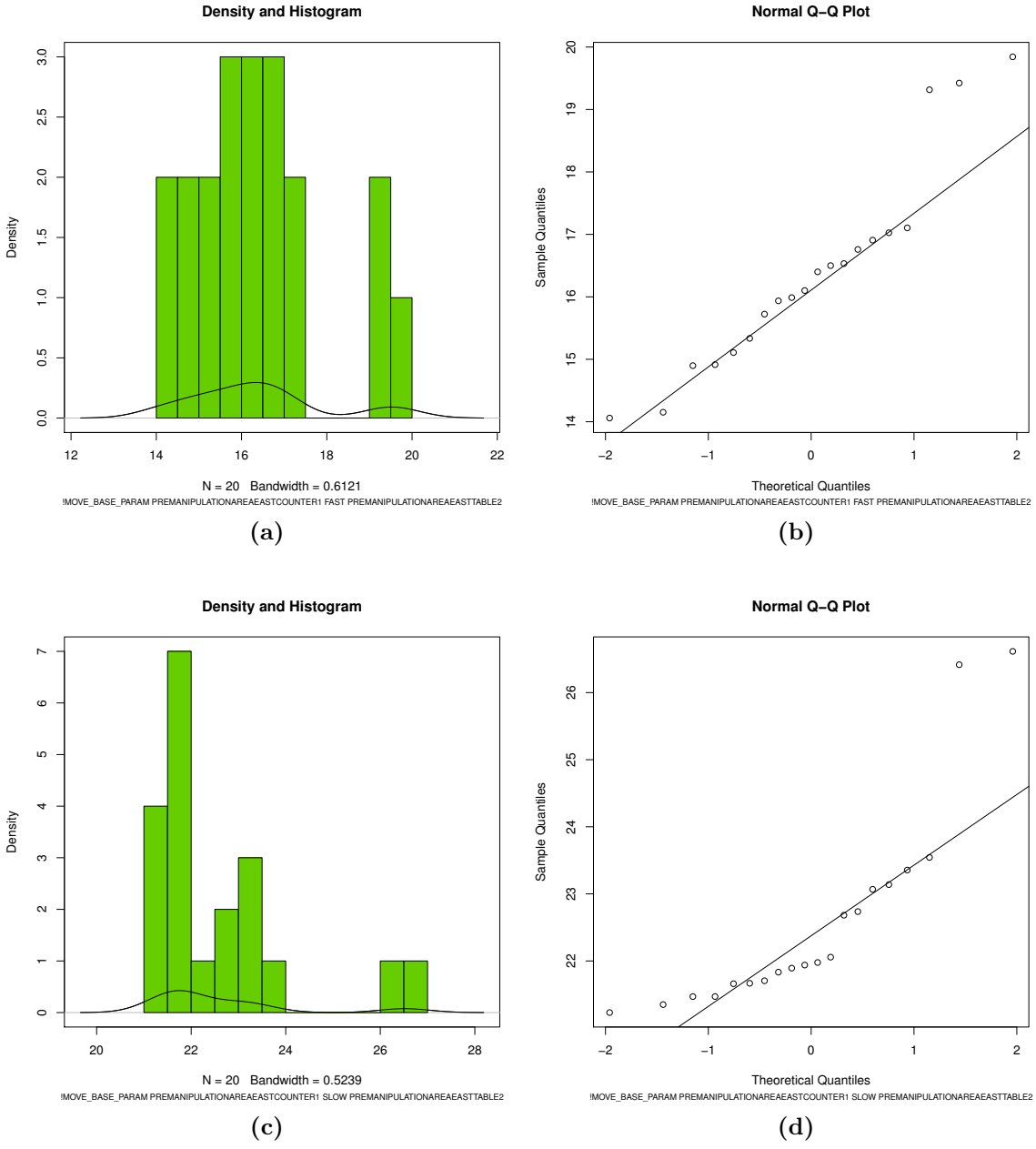


Fig. A.25: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

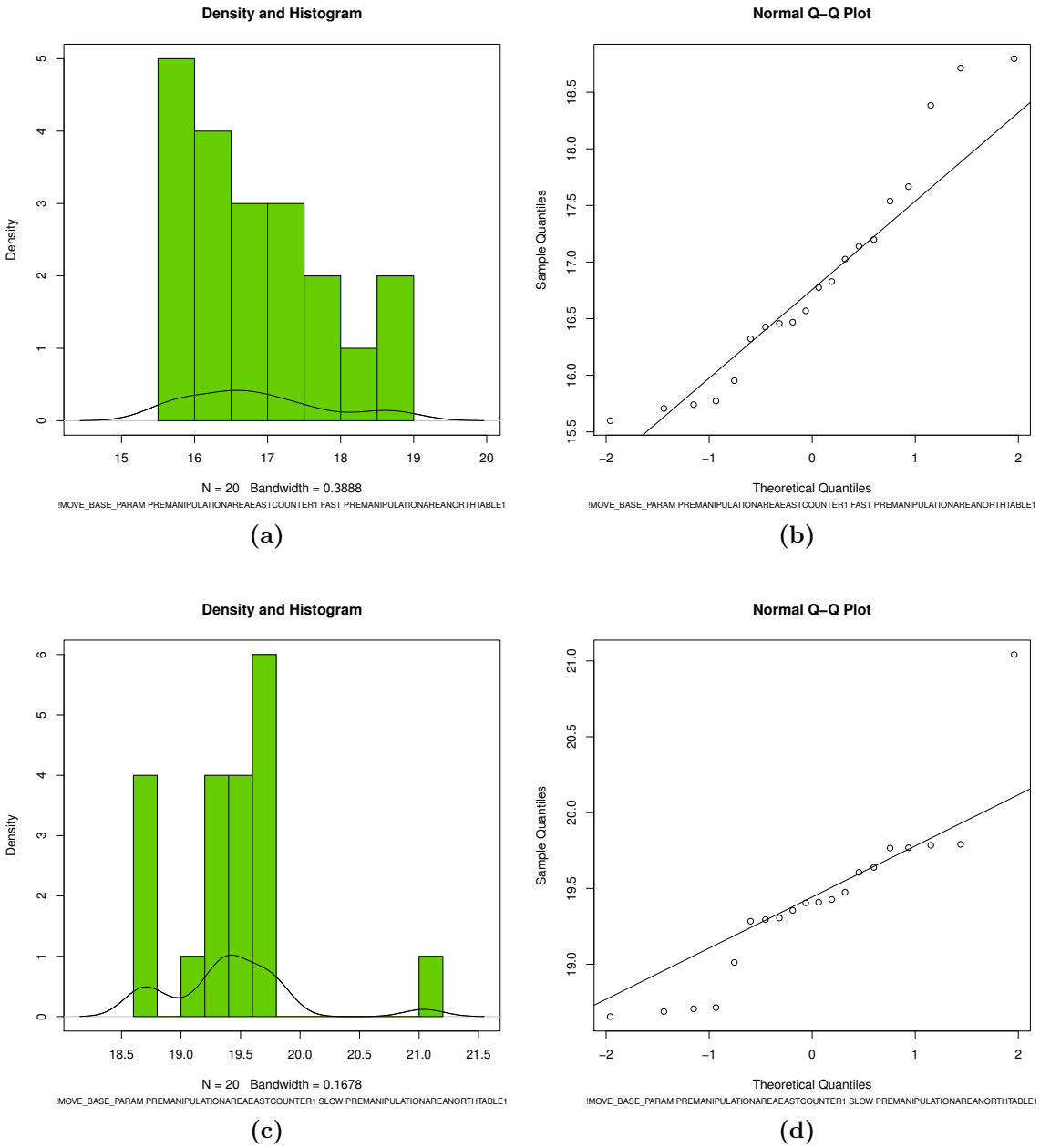


Fig. A.26: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

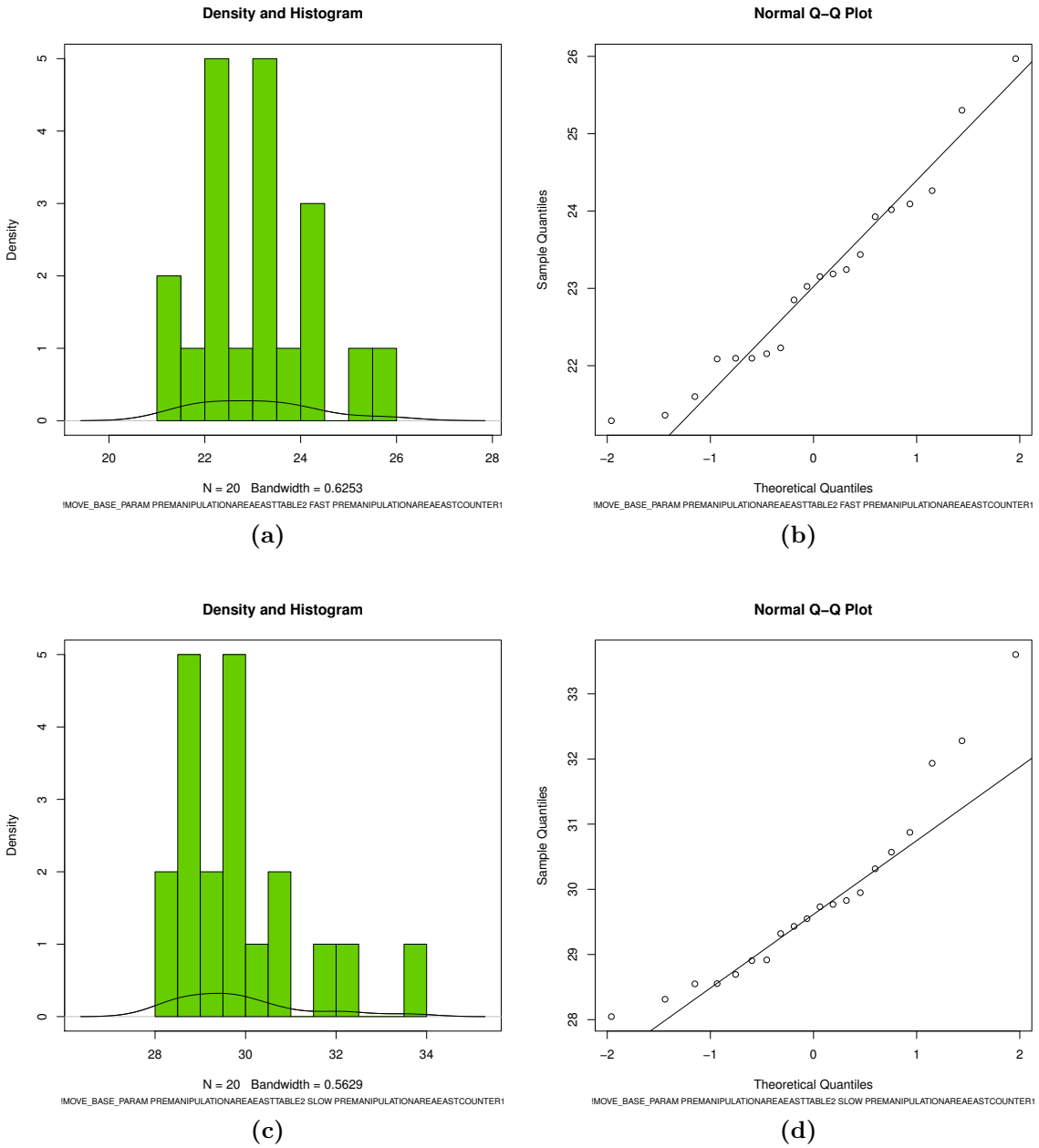


Fig. A.27: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

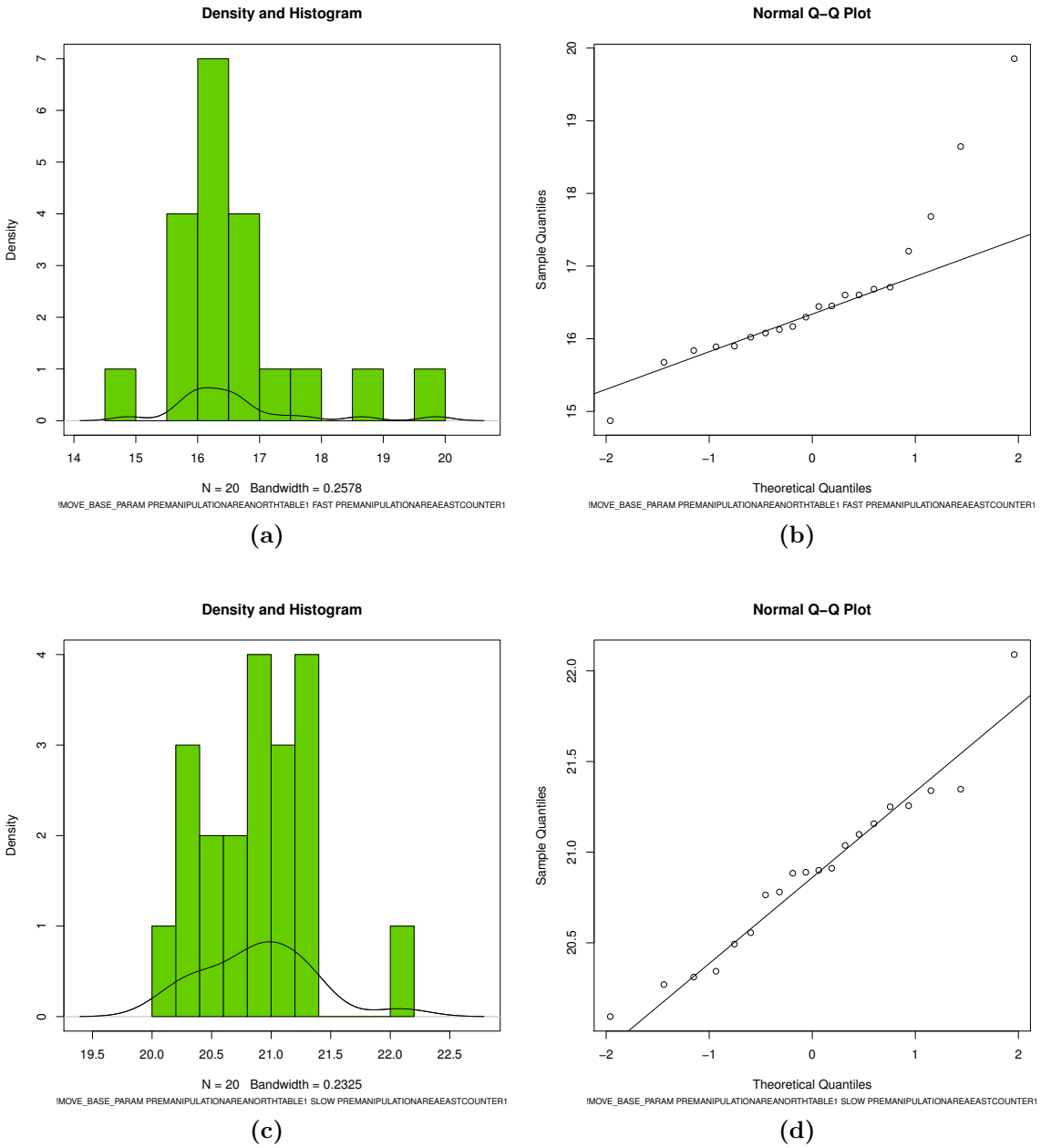


Fig. A.28: Statistics of action (a,c) Density and Histogram. (b,d) Q-Q Plot.

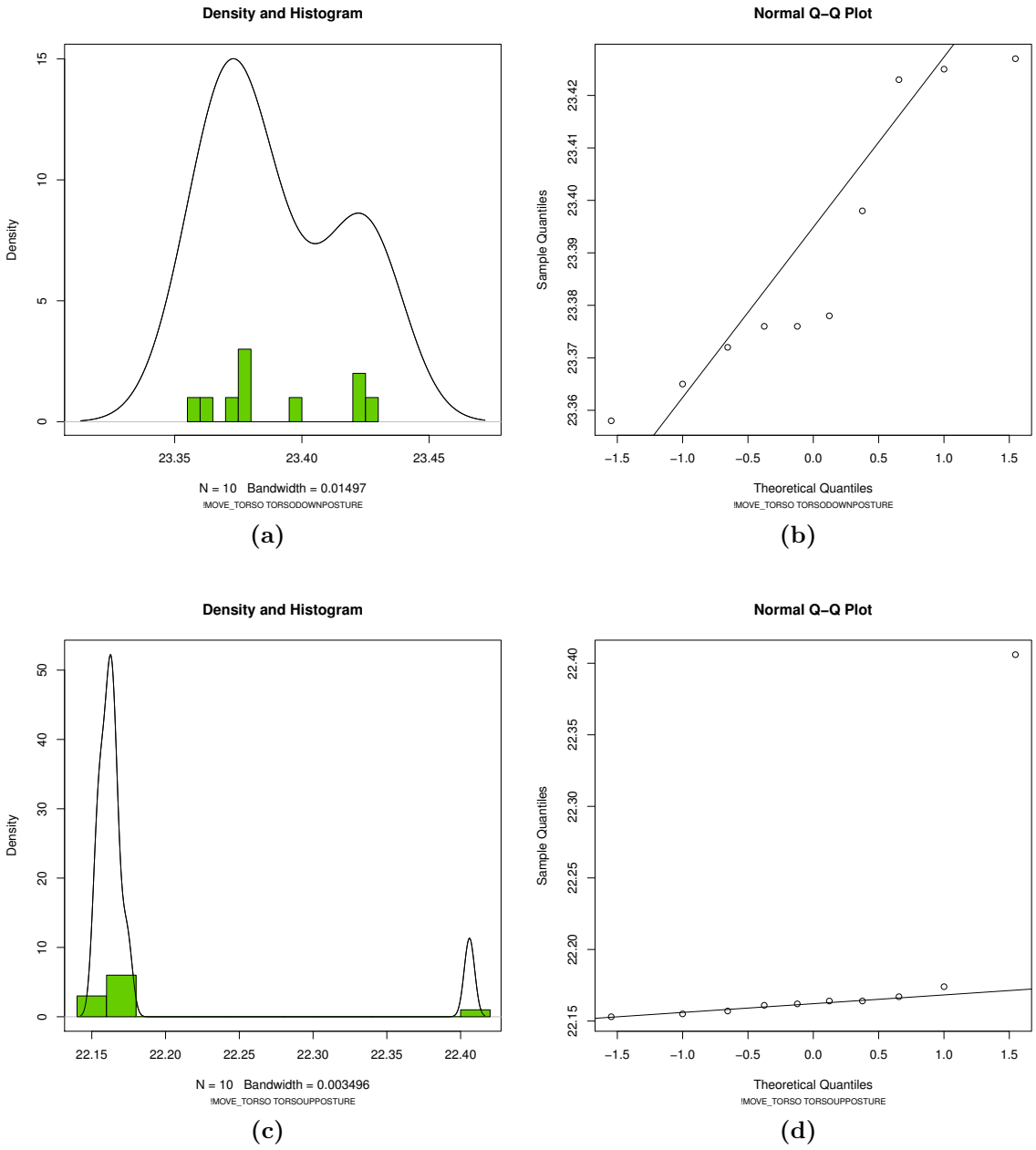


Fig. A.29: Statistics of action (a) Density and Histogram. (b) Q-Q Plot.

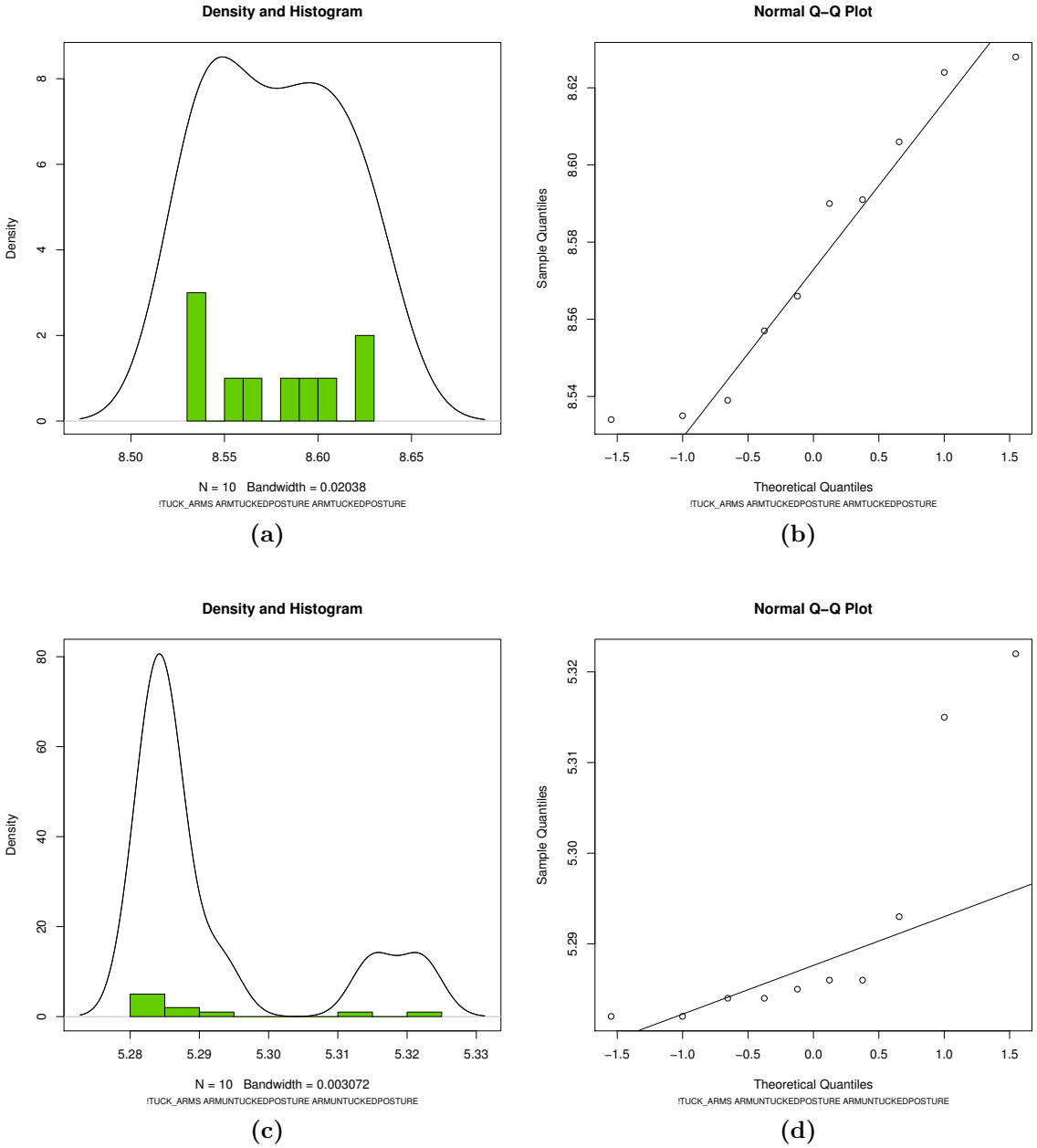


Fig. A.30: Statistics of action (a) Density and Histogram. (b) Q-Q Plot.

Bibliography

A

[AKPL13] N. Akhtar, A. Küstenmacher, P. Plöger, and G. Lakemeyer. Simulation-based approach for avoiding external faults. In *The 16th International Conference on Advanced Robotics (ICAR)*, 2013.

[All84] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Bal00] J. Baltes. A benchmark suite for mobile robots. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 1101–1106 vol.2, 2000.

[Bau94] Jean Baudrillard. *Simulacra and simulation*. Univ. Michigan Press, 1994.

[BB07] Adrian Boeing and Thomas Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 281–288, New York, NY, USA, 2007. ACM.

[BBF14] Byron Boots, Arunkumar Byravan, and Dieter Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[Bee00] Michael Beetz. *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*. Springer-Verlag, Berlin, Heidelberg, 2000.

[BGMH+12] Thomas Breuer, Geovanny R. Giorgana Macedo, Ronny Hartanto, Nico Hochgeschwender, Dirk Holz, Frederik Hegger, Zha Jin, Christian Müller, Jan Paulus, Michael Reckhaus, José Antonio Álvarez Ruiz, Paul G. Plöger, and Gerhard K. Kraetzschmar. Johnny: An autonomous service robot for domestic environments. *J. Intell. Robotics Syst.*, 66(1-2):245–272, April 2012.

[BHH04] Thorsten Belker, Martin Hammel, and Joachim Hertzberg. Plan projection under the appeal robot control architecture. In *8th Conf. Intelligent Autonomous Systems (IAS-8)*, pages 809–817, Amsterdam, The Netherlands, 2004. IOS Press.

[BHKLP12] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Manipulation with multiple action types. In *International Symposium on Experimental Robotics*, June 2012.

[BKLP13] J. Barry, L.P. Kaelbling, and T. Lozano-Perez. A hierarchical approach to manipulation with diverse actions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1799–1806, May 2013.

[Blo81] Ned Block. *Imagery (Bradford Books)*. MIT Press (MA), 1981.

[Blo83] Ned Block. Mental pictures and cognitive science. *Philosophical Review*, 93(October):499–542, 1983.

[BRGJ+11] J. Bohren, R.B. Rusu, E. Gil Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5568–5575. IEEE, 2011.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[BSD73] Neill Graham Bryce Seligman DeWitt, editor. *The Many-Worlds Interpretation of Quantum Mechanics*. Princeton Series in Physics. Princeton University Press, 1973. ISBN 0-691-08131-X.

[BW97] David Baraff and Andrew Witkin. Physically based modeling: Principles and practice. Carnegie Mellon University, 1997.

[Cas71] Edward S. Casey. Imagination: Imagining and the image. *Philosophy and Phenomenological Research*, 31(4):pp. 475–490, 1971.

[CMW09] I.Y.-H. Chen, B. MacDonald, and B. Wunsche. Mixed reality simulation for mobile robots. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 232–237, May 2009.

[CNSV14] Anthony G. Cohn, Bernd Neumann, Alessandro Saffiotti, and Markus Vincze. Robots learning from experiences (dagstuhl seminar 14081). In *Dagstuhl Reports*, volume 4, pages 97,107–108, 2014.

[Cra43] Kenneth J. W. Craik. *The Nature of Explanation*. Cambridge University Press, 1943.

[DDO+14] K. Dubba, P. Duckworth, M. A. Omari, J. Tayyub, Y. Gatsoulis, A. G. Cohn, D. C. Hogg, L. Hotz, B. Neumann, S. Rockel, L. S. Lopes, M. Oliveira, G. H. Lim, S. H. Kasaei, and V. Mokhtari. Conceptualising environment activities and events. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, November 2014.

[Deu85] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London*, 400:97–117, 1985.

[DHCS12] Mehmet Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. Physics-based grasp planning through clutter. In *Robotics: Science and Systems VIII*, July 2012.

[DHN13] Christian Dornhege, Andreas Hertle, and Bernhard Nebel. Lazy evaluation and subsumption caching for search-based integrated task and motion planning. In *Workshop at the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[dSGPA14] Lavindra de Silva, Mamoun Gharbi, Amit Kumar Pandey, and Rachid Alami. A new approach to combined symbolic-geometric backtracking in the context of human-robot interaction. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[dSPA13] Lavindra de Silva, Amit Kumar Pandey, and Rachid Alami. An interface for interleaved symbolic-geometric planning and backtracking. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[EKRZ13] Lasse Einig, Denis Klimentjew, Sebastian Rockel, and Jianwei Zhang. Parallel plan execution and re-planning on a mobile robot using state machines with htn planning systems. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, December 2013.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FBDT99] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *In Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 343–349, 1999.

[FBT97] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, Mar 1997.

[FCK+09] Jonathan Fink, Tom Collins, Vijay Kumar, Yasamin Mostofi, John Baras, and Brian Sadler. A simulation environment for modeling and development of algorithms for ensembles of mobile microsystems. In *SPIE Defense, Security, and Sensing*, volume 7318, pages 73180N–73180N–10. International Society for Optics and Photonics, 2009.

[FN71] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[GNHK06] David Gamez, Richard Newcombe, Owen Holland, and Rob Knight. Two simulation tools for biologically inspired virtual robotics. *Proceedings of the IEEE 5th chapter conference on advances in cybernetic systems, Sheffield*, pages 85–90, 2006.

[Gui08] Meen-Wah Gui. The basics of noise detection and filtering for borehole drilling data. In *The Open Civil Engineering Journal*, volume 2, pages 113–120, Taipei 106, Taiwan, 2008. Department of Civil Engineering, National Taipei University of Technology.

[GVH03] Brian P. 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)*, Coimbra, Portugal, June 2003.

[HC11] A. Harris and J. M. Conrad. Survey of popular robotics simulators, frameworks, and toolkits. In *IEEE Southeastcon*, pages 243–249, 2011.

[HIvdZ13] Dirk Holz, Luca Iocchi, and Tijn van der Zant. Benchmarking intelligent service robots through scientific competitions: The robocup@home approach. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford (USA), March 25-27 2013.

[HZZ+14] Joachim Hertzberg, Jianwei Zhang, Liwei Zhang, Sebastian Rockel, Bernd Neumann, Jos Lehmann, Krishna S. R. Dubba, Anthony G. Cohn, Alessandro Saffiotti, Federico Pecora, Masoumeh Mansouri, Štefan Konečný, Martin Günther, Sebastian Stock, Luis Seabra Lopes, Miguel Oliveira, Gi Hyun Lim, Hamidreza Kasaei, Vahid Mokhtari, Lothar Hotz, and Wilfried Bohlken. The race project. *KI - Künstliche Intelligenz*, 2014.

[ILSKa05] I. Iossifidis, G. Lawitzky, and R. Zöllner S. Knoop and. *Towards Benchmarking of Domestic Robotic Assistants*, volume 14 of *Advances in Human-Robot Interaction*. Springer-Verlag, Heidelberg, 2005.

[JL83] Philip N. Johnson-Laird. *Mental Models*. Harvard University Press, 1983.

[JL95] Philip N. Johnson-Laird. Mental models, deductive reasoning, and the brain. In *The Cognitive Neurosciences*, pages 999–1008. Mit Press, 1995.

[JLS12] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. In *ICML*, 2012.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KDB11] Lars Kunze, Mihai Emanuel Dolha, and Michael Beetz. Logic programming with simulation-based temporal projection for everyday robot object manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September, 25–30 2011. Best Student Paper Finalist.

[KDGB11] Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, and Michael Beetz. Simulation-based temporal projection of everyday robot object manipulation. In Yolum, Tumer, Stone, and Sonenberg, editors, *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Taipei, Taiwan, May, 2–6 2011. IFAAMAS.

[KEP05] Danny M. Kaufman, Timothy Edmunds, and Dinesh K. Pai. Fast frictional dynamics for rigid bodies. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 946–956, New York, NY, USA, 2005. ACM.

[KGT01] S. M. Kosslyn, G. Ganis, and W. L. Thompson. Neural foundations of imagery. In *Nature Reviews: Neuroscience*, 2001.

[KH04] N. Koenig and A Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, Sept 2004.

[KHB12] Lars Kunze, Andrei Haidu, and Michael Beetz. Making virtual pancakes — acquiring and analyzing data of everyday manipulation tasks through interactive physics-based simulations. In *Poster and Demo Track of the 35th German Conference on Artificial Intelligence (KI-2012)*, Saarbrücken, Germany, September 24–27 2012.

[Kli15] Denis Klimentjew. *Szenenanalyse in unstruktuierter Umgebung Adaptives Verfahren zur Objekterkennung basierend auf der Multi-Sensor-Fusion und aktiven Wahrnehmung*. PhD thesis, Universität Hamburg, 2015.

[KLP12] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated robot task and motion planning in belief space, 2012.

[KLP+14] Peter Kaiser, Mike Lewis, Ronald P. A. Petrick, Tamim Asfour, and Mark Steedman. Extracting common sense knowledge from text for robot planning. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[Kon00] K. Konolige. A gradient method for realtime robot control. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 639–646 vol.1, 2000.

[KRZ12] Denis Klimentjew, Sebastian Rockel, and Jianwei Zhang. Towards scene analysis based on multi-sensor fusion, active perception and mixed reality in mobile robotics. In *Proceedings of the IEEE First International Conference on Cognitive Systems and Information Processing (CSIP2012)*, 2012.

[KRZ13] Denis Klimentjew, Sebastian Rockel, and Jianwei Zhang. Active scene analysis based on multi-sensor fusion and mixed reality on mobile systems. *Foundations and Practical Applications of Cognitive Systems and Information Processing (Chapter 69), Advances in Intelligent Systems and Computing*, 215:795–810, 2013.

[KS13] Hema Swetha Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Robotics: Science and Systems*, 2013.

[KSPS14] Štefan Konečný, Sebastian Stock, Federico Pecora, and Alessandro Saffiotti. Planning domain + execution semantics: a way towards robust execution? In *Proc. of Qualitative Representations for Robots, AAAI Spring Symposium*, 2014.

[KyHR03] Nikolaos Mavridis Kai-yuh Hsiao and Deb Roy. Coupling perception and simulation: Steps towards conversational robotics. In *The Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[LB13] Daniel Leidner and Christoph Borst. Hybrid reasoning for mobile manipulation based on object knowledge. In *Workshop at the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[LC01] L.S. Lopes and J.H. Connell. Semisentient robots: routes to integrated intelligence. *Intelligent Systems, IEEE*, 16(5):10–14, Sep 2001.

[LDS+14] Daniel Leidner, Alexander Dietrich, Florian Schmidt, Christoph Borst, and Alin Albu-Schäffer. Object-centered hybrid reasoning for whole-body mobile manipulation. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[Loa01] Brian J. Loasby. Cognition, imagination and institutions in demand creation. *Journal of Evolutionary Economics*, 11:7–21, 2001.

[Lod96] Janice E. Lodato. Moral imagination: Implications of cognitive science for ethics. *Dialogue: Canadian Philosophical Review/Revue canadienne de philosophie*, 35:204–207, December 1996.

[MB09] 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)*, 2009.

[MEBF+10] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[MHN08] Hugo Gravato Marques, Owen Holland, and Richard Newcombe. A modelling framework for functional imagination. In *AISB Convention*, pages 51–58, 2008.

[MKNH08] Hugo Gravato Marques, Rob Knight, Richard Newcombe, and Owen Holland. An anthropomimetic robot with imagination: one step closer to machine consciousness? In *Nokia Workshop on Machine Consciousness*, 2008.

[ML09] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[MP06] M.D. Moffitt and M.E. Pollack. Optimal rectangle packing: A meta-CSP approach. In *Proc. of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS)*, 2006.

[NHRL14] Bernd Neumann, Lothar Hotz, Pascal Rost, and Jos Lehmann. A robot waiter learning from experiences. *Machine Learning and Data Mining in Pattern Recognition*, LNCS 8556:285–299, 2014.

[NIK+03] Dana Nau, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[NMAC+01] Dana Nau, Héctor Muñoz-Avila, Yue Cao, Amnon Lotem, and Steven Mitchell. Total-order planning with partially ordered subtasks. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, pages 425–430, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Nor02] Donald A. Norman. *The Design of Everyday Things*. Basic Books, Inc., New York, NY, USA, 2002.

[OFY07] Fumitaka Otsuka, Hikari Fujii, and Kazuo Yoshida. Development of three dimensional dynamics simulator with omnidirectional vision model. In Gerhard Lakemeyer, Elizabeth Sklar, DomenicoG. Sorrenti, and Tomoichi Takahashi, editors, *RoboCup 2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Computer Science*, pages 523–531. Springer Berlin Heidelberg, 2007.

[OSB14] Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[PMvRH13] Federico Pecora, Masoumeh Mansouri, Stephanie von Riegen, and Lothar Hotz. Deliverable d1.2 - software framework for hybrid knowledge representation and reasoning. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, 6 2013.

[Pra15] Gill A. Pratt. Is a cambrian explosion coming for robotics? *Journal of Economic Perspectives*, 29(3):51–60, 2015.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully B. Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[RKZ12] Sebastian Rockel, Denis Klimentjew, and Jianwei Zhang. A multi-robot platform for mobile robots - a novel evaluation and development approach with multi-agent technology. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 470–477, Hamburg, Germany, 2012. University of Hamburg.

[RKZZ14] Sebastian Rockel, Denis Klimentjew, Liwei Zhang, and Jianwei Zhang. An hyperreality imagination based reasoning and evaluation system (hires). In *IEEE/RSJ International Conference on Robots and Automation (ICRA)*, Hong Kong (China), June 2014.

[RM09] Elena Messina Raj Madhavan, Edward Tunstel. *Performance Evaluation and Benchmarking of Intelligent Systems*. Springer, 2009. ISBN: 978-1-4419-0491-1 (Print) 978-1-4419-0492-8 (Online).

[RN07] Jochen Renz and Bernhard Nebel. Qualitative spatial reasoning using constraint calculi. In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.

[RNZ+13] S. Rockel, B. Neumann, J. Zhang, K. S. R. Dubba, A. G. Cohn, Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, S. Stock, J. Hertzberg, A. M. Tomé, A. J. Pinho, L. S. Lopes, S. von Riegen, and L. Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford (USA), March 2013.

[SGH14] Sebastian Stock, Martin Günther, and Joachim Hertzberg. Generating and executing hierarchical mobile manipulation plans. In *Proc. of ISR/Robotik*, 2014.

[SHL+13] S. Stock, J. Hertzberg, J. Lehmann, B. Neumann, W. Bohlken, L. Hotz, S. Rockel, and Š. Konečný. Deliverable d4.2 - consolidated versions of prediction, expectation, and plan execution. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, 11 2013.

[SK08] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*, chapter 17. Springer, 2008.

[SP97] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[Teg98] Max Tegmark. The interpretation of quantum mechanics: Many worlds or many words? In *Fortsch.Phys.*, 1998. arXiv:quant-ph/9709032.

[Tho14] Nigel J.T. Thomas. Mental imagery. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, spring 2014 edition, 2014.

[TT01] Nobuyoshi Terashima and John Tiffin. *HyperReality: Paradigm for the Third Millenium*. Routledge, 2001.

[Tul72] Endel Tulving. *Organization of Memory*, chapter Episodic and Semantic Memory, pages 381–402. Academic Press, New York, 1972.

[vB90] Peter van Beek. *Exact and approximate reasoning about qualitative temporal relations*. PhD thesis, University of Waterloo, Waterloo, Ont., Canada, Canada, 1990. UMI Order No. GAXNN-61098.

[vdZI11] Tijn van der Zant and Luca Iocchi. Robocup@home: Adaptive benchmarking of robot bodies and minds. In Bilge Mutlu, Christoph Bartneck, Jaap Ham, Vanessa Evers, and Takayuki Kanda, editors, *Social Robotics*, volume 7072 of *Lecture Notes in Computer Science*, pages 214–225. Springer Berlin Heidelberg, 2011.

[VMJB13] J.G. Victores, S. Morante, A. Jardon, and C. Balaguer. Towards robot imagination through object feature inference. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5694–5699, Nov 2013.

[WBVdLS08] K.A. Wyrobek, E.H. Berger, H.F.M. Van der Loos, and J.K. Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2165–2170, May 2008.

[WHR10] Erik Weitnauer, Robert Haschke, and Helge Ritter. Evaluating a physics engine as an ingredient for physical reasoning. In Noriaki Ando, Stephen Balakirsky, Thomas Hemker, Monica Reggiani, and Oskar Stryk, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472 of *Lecture Notes in Computer Science*, pages 144–155. Springer Berlin Heidelberg, 2010.

[Wis09] Thomas Wisspeintner. *Prototyping and benchmarking autonomous mobile service robot applications*. PhD thesis, University of Berlin, June 2009.

[WvdZIS09] Thomas Wisspeintner, Tijn van der Zant, Luca Iocchi, and Stefan Schiffer. Robocup@home: Scientific competition and benchmarking for domestic service robots. *Interaction Studies*, 10(3):392–426, 2009.

[WvdZIS10] Thomas Wisspeintner, Tijn van der Zant, Luca Iocchi, and Stefan Schiffer. Robocup@home: Results in benchmarking domestic service robots. In Jacky Baltes, Michail G. Lagoudakis, Tadashi Naruse, and Saeed Shiry Ghidary, editors, *RoboCup 2009*, chapter

RoboCup@Home: results in benchmarking domestic service robots, pages 390–401. Springer-Verlag, Berlin, Heidelberg, 2010.

[ZRP+12] Liwei Zhang, Sebastian Rockel, Federico Pecora, Luís Seabra Lopes, Alessandro Saffiotti, and Bernd Neumann. Deliverable d5.1 - evaluation infrastructure. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, November 2012.

[ZRP+13] Liwei Zhang, Sebastian Rockel, Federico Pecora, Lothar Hotz, Zhenli Lu, and Jianwei Zhang. Evaluation metrics for an experience based mobile artificial cognitive system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo (Japan), 2013.

[ZRZ13] Liwei Zhang, Sebastian Rockel, and Jianwei Zhang. Exception handling for experience-based mobile cognitive systems in restaurant environments exemplified by guest detection. In *IEEE International Conference on Information and Automation (ICIA)*, Yinchuan (China), 2013.

[ZV10] S. Zickler and M. Veloso. Variable level-of-detail motion planning in environments with poorly predictable bodies. In *19th European Conference on Artificial Intelligence, ECAI 2010*, August 2010.

Publications

The following publications either resulted from this thesis or include significant results from the thesis. The list is ordered by date.

- Sebastian Rockel, Štefan Konečný, Sebastian Stock, Joachim Hertzberg, Federico Percora, Jianwei Zhang, *Integrating physics-based Prediction with Semantic Plan Execution Monitoring*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015. (accepted)
- Haiyang Jin, Liwei Zhang, Sebastian Rockel, Jun Zhang, Ying Hu, Jianwei Zhang, *Optical Tracking based Tele-control System for Tabletop Object Manipulation Tasks*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015. (accepted)
- J. Hertzberg, J. Zhang, L. Zhang, S. Rockel, B. Neumann, J. Lehmann, K. S. R. Dubba, A. G. Cohn, A. Saffiotti, F. Pecora, M. Mansouri, Š. Konečný, M. Günther, S. Stock, L. S. Lopes, M. Oliveira, G. H. Lim, H. Kasaei, V. Mokhtari, L. Hotz, W. Bohlken, *The RACE Project*, KI - Künstille Intelligenz Journal, Springer Berlin Heidelberg, 2014.
- Anthony G. Cohn, Bernd Neumann, Alessandro Saffiotti, Markus Vincze, *Robots Learning from Experiences (Dagstuhl Seminar 14081)*, Dagstuhl Reports, vol. 4, num. 2, 2014, Dagstuhl, (Germany).
- Sebastian Rockel, Denis Klimentjew, Liwei Zhang, Jianwei Zhang, *An Hyperreality Imagination based Reasoning and Evaluation System (HIRES)*, 2014 IEEE International conference on Robotics and Automation (ICRA 2014), Hong Kong (China), 2014.
- Sebastian Rockel, *Predicting Robot Action Results physically correct: Towards Imaginary Planning*, Dagstuhl Seminar Feb 16-21, 2014, Robots Learning from Experiences, Wadern (Germany).
- Liwei Zhang, Sebastian Rockel, Alessandro Saffiotti, Federico Pecora, Lothar Hotz, Zhenli Lu, Denis Klimentjew, Jianwei Zhang, *Evaluation Metrics for an Experience-based Mobile Artificial Cognitive System*, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Workshop on Metrics of Embodied Learning Processes in Robots and Animals, Tokyo (Japan), 2013.
- Liwei Zhang, Sebastian Rockel, Jianwei Zhang, *Exception Handling for Experience-based Mobile Cognitive Systems in Restaurant Environments Exemplified by Guest Detection*, IEEE International Conference on Information and Automation (ICIA), August 2013, Yinchuan (China).

- Lasse Einig, Denis Klimentjew, Sebastian Rockel, Jianwei Zhang, *Parallel Plan Execution and Re-planning on a Mobile Robot using State Machines with HTN Planning Systems*, IEEE International Conference on Robotics and Biomimetics (ROBIO), December 2013, Shenzhen (China).
- S. Rockel, B. Neumann, J. Zhang, K. S. R. Dubba, A. G. Cohn, Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, S. Stock, J. Hertzberg, A. M. Tomé, A. J. Pinho, L. S. Lopes, S. von Riegen, L. Hotz, *An Ontology-based Multi-level Robot Architecture for Learning from Experiences*, In: Proc. Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium, March 25-27, Stanford (USA), 2013.
- Denis Klimentjew, Sebastian Rockel, Jianwei Zhang, *Active Scene Analysis Based on Multi-Sensor Fusion and Mixed Reality on Mobile Systems*, Foundations and Practical Applications of Cognitive Systems and Information Processing (Chapter 69), Advances in Intelligent Systems and Computing, Vol. 215, Softcover ISBN 978-3-642-37834-8, eBook ISBN 978-3-642-37835-5, Springer 2013.
- Denis Klimentjew, Sebastian Rockel, Jianwei Zhang, *Towards Scene Analysis based on Multi-Sensor Fusion, Active Perception and Mixed Reality in Mobile Robotics*, In Proceedings of the IEEE First International Conference on Cognitive Systems and Information Processing (CSIP2012), to be held on Dec. 15-17, 2012 in Beijing, China, 2012.
- Sebastian Rockel, Denis Klimentjew, Jianwei Zhang, *A Multi-Robot Platform for Mobile Robots - A Novel Evaluation and Development Approach with Multi-Agent Technology*, In Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), University of Hamburg, Hamburg, Germany, 2012.

Glossary

The following bibliographical references are alphabetically sorted according to the last name of the first author.

2D two dimensional. 3, 4, 12, 28, 33, 38, 41, 53, 64

3D three dimensional. 4, 28, 33, 36, 41, 52, 53, 58, 59, 78

AI artificial intelligence. 17, 95

AMCL Adaptive Monte Carlo Localization. 33, 54, 57, 59

API Application Programming Interface. 19, 36, 38, 39

ASEF Automated Scenario Execution Framework. 61, 93

C1 PR2 on-board computer 1. 26

C2 PR2 on-board computer 2. 26

CAD Computer-Aided Design. 41, 78

CLI Command Line Interface. 79

CPU Central Processing Unit. 25, 53, 54, 55, 71

DAMA diverse action manipulation. 13

DARPA Defense Advanced Research Projects Agency. 22

DOF degree-of-freedom. 18, 24

DWA Dynamic Window Approach. 33, 42

EBCL example-based compositional learning. 83

EU European Union. 5, 20, 35

EURON European Robotics Research Network. 21

FIM Fitness to Ideal Model. 21

FP7 Framework Programme for Research and Technological Development. 5, 35

FPS frames per second. 28

GB Gigabyte. 25, 26, 53

GLUT the OpenGL Utility Toolkit. 38

GNU GNU's Not Unix!. 52

GPU Graphics Processing Unit. 39, 40

GTP geometric task planner. 10

GUI Graphical User Interface. 37, 38, 39, 55

HD high definition. 28

HDD hard disk drive. 25

HIRES Hyperreality Imagination-based Reasoning and Evaluation System. 1, 49, 60, 64, 78, 80

HPN Hierarchical Planning in the Now. 13

HTN Hierarchical Task Network. 3, 9, 10, 13, 14, 18, 48, 92

HW Hardware. 25, 31

ID Identifier. 51, 58, 59

IDF Iterative Distance Fitter. 41

IMU Inertial Measurement Units. 28, 29, 33

JNI Java Native Interface. 19

KR knowledge representation. 9

LED Light-emitting diode. 28

Meta-CSP Meta-Constraint Satisfaction Problems. 9

ML machine learning. 15

MP Mega-pixel. 28

MR Mixed Reality. 1

MRS Multi-Robot System. 19

MWI Many Worlds Interpretation. 68, 69

ODE Open Dynamics Engine. 14, 15, 35, 38, 39, 40, 72, 88, 94

OGRE Object-Oriented Graphics Rendering Engine. 38

OpenCV Open source Computer Vision library. 36

OpenGL the Open Graphics Library. 38

ORK Object Recognition Kitchen. 41

OS Operating System. 31, 72, 73, 88

OSRF Open Source Robotics Foundation, Inc.. 33

PCL Point Cloud Library. 36

PDDL Planning Domain Definition Language. 13

PMA pre-manipulation area. 65, 66, 71, 73, 77, 81, 85, 87

PR1 Personal Robot 1. 23

PR2 Personal Robot 2. 10, 12, 17, 23, 24, 25, 26, 28, 29, 31, 33, 36, 37, 41, 42, 49, 51, 53, 60, 61, 65, 71, 73, 91, 95

PROLOG A general purpose logic programming language. 14

PSR Predictive State Representation. 18

QQ plot A Quantile-Quantile plot is a probability plot indicating either a standard normal distribution of the underlying data or rejecting it. 73

QT4 "Cute" cross-platform programming library. 37, 38, 39

RACE Robustness by Autonomous Competence Enhancement. 5, 33, 35, 42, 46, 47, 48, 49, 53, 54, 55, 60, 61, 62, 69, 71, 73, 79, 91, XI

RAM Random Access Memory. 26

RANSAC Random Sample Consensus. 33, 41

RGB red, green and blue color space. 28, 61

RGB-D Color and Depth. 18, 36, 58

ROS Robot Operating System. 17, 19, 20, 23, 31, 33, 35, 36, 37, 38, 39, 41, 42, 46, 48, 50, 51, 54, 78, 91

RPC Remote Procedure Call. 36

RRT Rapidly-exploring Random Trees. 12

Rviz ROS visualization. 55

SDF Simulation Description Format. 58

SEM Semantic Execution Monitor. 47, 48, 61, 84, 86

SHOP2 Simple Hierarchical Ordered Planner 2. 13, 48, 60

SIMNOS A real-time physics engine-based simulation. 16

SLAM Simultaneous Localization And Mapping. 33

SMACH State MACHine. 13, 14

SPA Sense-Plan-Act. 45, 92

STL Surface Tesselation Language. 38, 57

STRIPS STanford Research Institute Problem Solver. 9

TAMS Technical Aspects of Multimodal Systems. 23, 33, 54

TB Terabyte. 26

URDF Unified Robot Description Format. 58

USC University of Southern California. 37

VLOD Variable Level-Of-Detail. 12

WAP wireless access point. 31

WiFi local area wireless technology. 31

XML Extensible Markup Language. 37, 38, 58

List of Figures

List of Tables

List of Algorithms

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 10. September 2015

Sebastian Rockel