Belief-Space Planning for Resourceful Manipulation and Mobility

Dirk Ruiken

BELIEF-SPACE PLANNING FOR RESOURCEFUL MANIPULATION AND MOBILITY

A Dissertation Presented

by

DIRK RUIKEN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2017

College of Information and Computer Sciences

BELIEF-SPACE PLANNING FOR RESOURCEFUL MANIPULATION AND MOBILITY

A Dissertation Presented

by

DIRK RUIKEN

DEDICATION

To my family.

ACKNOWLEDGMENTS

Looking back onto my time as a Ph.D. student, I feel very fortunate as I really enjoyed it here. When I first came to UMass as an exchange student, I had the opportunity to work with Rod Grupen on the uBot project—version 4 back then. I enjoyed working with him and on uBot(s) so much that I came back to pursue my Ph.D. I want to thank Rod for continuing to be a great advisor. Through all these years he has supported and guided me in my research, and I have never lost the joy of it. Thank you, Rod, for these great years.

I also would like to thank the members of my committee, Shlomo Zilberstein, Erik Learned-Miller, and Frank Sup, for their insight and support. Frank's support for the development of the latest uBot was invaluable.

An important part of my time as a graduate student was the great work environment with the lab members in the Laboratory for Perceptual Robotics (LPR) and the Autonomous Mobile Robotics Laboratory (AMRL). LPR: Tiffany Liu, Li Ku, Jay Ming Wong, Mitchell Hebert, Takeshi Takahashi, Scott Jordan, Mike Lanighan, Uday Savaria, Kushal Sahare, Eric Wilkinson, Scott Kuindersma, Shiraj Sen, Grant Sherrick, Steve Hart, Ed Hannigan, Patrick Deegan, Bryan Thibodeau, Dan Xie, Yun Lin, Chao Ou, Hee-Tae Jung, Tom Billings, Dubi Katz, and Emily Horrell. AMRL: Joydeep Biswas, Samer Nashed, Jarrett Holtz, Spencer Lane, Sadegh Rabiee, David Balaban, Edward Schneeweiss, and Kyle Vedder.

Many thanks have to go to Leeanne Leclerc for always looking out for us and making sure that we stay on track to graduate. Both Priscilla Scott and Laurie Downey have made my life much easier and provided fantastic administrative support despite my best efforts to order a shear endless number of robot parts. Likewise, the international programs office at UMass provided great support, but also many friends. I have participated in and helped organize their orientation and welcome weeks for several years, and they have been always delighting.

A special thanks has to go to the original uBot-4/5 team that was a part of the reason I came to UMass and made my first years special: Patrick Deegan, Bryan Thibodeau, Ed Hannigan, and Scott Kuindersma. Working with Ed and Scott was especially fantastic—I could never wish for better coworkers and team mates.

During all my time in Amherst, I have been very fortunate in the friends and housemates I have made. They have enriched my life and generally kept me happy and sane. I am sure that I missed someone, but I would like to thank George Konidaris, Bruno Ribero, Ed Hannigan, Scott Kuindersma, Philip Thomas, Shiraj Sen, Mitchell Hebert, Marc Cartright, Ilene Cartright, Orion Cartright, Sebastian Cartright, Sam Huston, Thomas Rossi, Bobby Simidchieva, Jan Panteli, Laura Sevilla, Katerina Marazopoulou, Bruno Castro da Silva, Yoonheui Kim, Armita Kaboli, Henry Feild, Jackie Feild, Sarah Osentoski, Megan Olsen, Tim Wood, Stefan Christof, Francesca Colantuoni, Maria Turrero, Michael Prokle, Alex Valencik, Jamie Klingensmith, Nisha Kini, Kevin Winner, Pinar Ozisik, Luis Pineda, Amanda Gentzel, Kaleigh Clary, Janet Guo, Su Lin Blodgett, Zeki Yalniz, Siddheshwari Advani, Garrett Bernstein, Stefan Christov, Huong Phan, Sophal Khun, Justin Svegliato, Myungha Jang, John Vilk, Elaine Kenseth, Lucy Fyffe, Lisa-Beth Waterworth, Kenzie Millar, Izzy Robins, Katerina Byanova, Anna Womack, Anna Heiler, Armando Leal, Marien Villafaña, Three Flower, and Gulag. You made graduate school truly enjoyable!

I would like to particularly acknowledge Tiffany Liu, Kyle Wray, and Samer Nashed. They have all been good friends, colleagues, and house mates. They were especially helpful during the final stages of this dissertation, always provided insightful feedback, and helped me tremendously in formulating my thoughts and research ideas.

Finally, I am very grateful to my family who was very understanding when I decided to pursue a Ph.D. on another continent. They have been nothing but supportive over all these years. I also have to thank my friends Bente Hampel, Leo Braun and Simone Braun-Herren, Patrik Lengerer, Julia Hochmuth, and Alejandra Villafaña who have made sure to remain a constant factor in my life no matter how far we are apart geographically. Thank you all!

ABSTRACT

BELIEF-SPACE PLANNING FOR RESOURCEFUL MANIPULATION AND MOBILITY

MAY 2017

DIRK RUIKEN

Diploma (M.Sc.), TECHNISCHE UNIVERSITÄT DARMSTADT

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic A. Grupen

Robots are increasingly expected to work in partially observable and unstructured environments. They need to select actions that exploit perceptual and motor resourcefulness to manage uncertainty based on the demands of the task and environment. The research in this dissertation makes two primary contributions. First, it develops a new concept in resourceful robot platforms called the UMass uBot and introduces the sixth and seventh in the uBot series. uBot-6 introduces multiple postural configurations that enable different modes of mobility and manipulation to meet the needs of a wide variety of tasks and environmental constraints. uBot-7 extends this with the use of series elastic actuators (SEAs) to improve manipulation capabilities and support safer operation around humans.

The resourcefulness of these robots is complemented with a belief-space planning framework that enables task-driven action selection in the context of the partially observable environment. The framework uses a compact but expressive state representation based on object models. We extend an existing affordance-based object model, called an aspect transition graph (ATG), with geometric information. This enables object-centric modeling of features and actions, making the model much more expressive without increasing the complexity. A novel task representation enables the belief-space planner to perform general object-centric tasks ranging from recognition to manipulation of objects. The approach supports the efficient handling of multi-object scenes.

The combination of the physical platform and the planning framework are evaluated in two novel, challenging, partially observable planning domains. The *ARcube domain* provides a large population of objects that are highly ambiguous. Objects can only be differentiated using multi-modal sensor information and manual interactions. In the *dexterous mobility domain*, a robot can employ multiple mobility modes to complete navigation tasks under a variety of possible environment constraints. The performance of the proposed approach is evaluated using experiments in simulation and on a real robot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

Increasingly, new applications for robots are being considered that remove them from the tightly controlled manufacturing setting and place them in unstructured, human environments. The unstructured environments generate a wide variety of constraints that cannot be anticipated ahead of time, and the number of tasks is constantly increasing and can vary a lot based on the situation.

Mobile manipulators are capable, theoretically, of dealing with such a variety of tasks and constraints in unstructured environments. These systems are often designed with excess degrees of freedom and with redundant sensors leading to the potential for sensory and motor flexibility. However, applications of these new machines require more than the *potential* for flexibility. So robots not only have to be resourceful and have many capabilities, but also need to be able to reason about the best approach to each problem based on the task constraints and the current environment.

In 1921, Wolfgang Köhler had chimpanzees solve a number of tasks with increasing difficulty [69]. For example, a banana would hang on a rope just out of jumping reach. The chimpanzee would try a couple of jumps, realize that it is too high, and quickly survey what other things could be used to help him. He would drag a box underneath the banana, thus increasing his jumping height. In subsequent experiments, the banana would be fixed even higher such that one box is not sufficient. The chimpanzee would realize after a few tries that the previous solution was not sufficient any more and get a second box. After several trials, the boxes are stacked, and the banana can be reached by jumping from the top of the stack. This example demonstrates the ability to realize when a previously successful strategy is no longer sufficient, to improve strategies, and to reason about which objects in the environment could be combined with own capabilities to find a potential strategy to solve a problem at hand. This combination of skills and adaptation is missing in state-of-the-art robotics and is critical to the successful application of robots in unstructured domains.

In order for a robot to succeed in situations that are not necessarily anticipated beforehand and for dealing with constraints imposed by different tasks, it needs to have a wide variety of skills at its disposal and needs to know how to use them.

1.1 Research Challenges

We identify three major challenges for success of robots in unstructured environments:

1. ***The robot should have the sensory and motor resourcefulness to be able to perform tasks with different functionally equivalent actions or sequences of actions.*** A robot is usually only tasked with things within its capabilities. However, slight variations of the task or the environment may render a single solution ineffective. In order to succeed in such situations a robot needs to have a wide variety of skills at its disposal and needs to know how to use them. Traditionally, these skills are either acquired autonomously in a constrained setting or pre-programmed.

2. ***The robot should be able to select actions based on the task and the environment to exploit perceptual and motor resourcefulness.*** This requires a good estimation of the state of the robot and of the environment around it. Unfortunately, the world is usually partially observable, and observations are uncertain, and thus the state often cannot be determined completely. Additionally, the robot needs to be able to predict the effects of its actions. Like

observations, action outcomes usually are uncertain. The robot has to factor the uncertainty in the state of the world and in action outcomes into his action selection.

3. ***The robot should be able to interact with objects or the environment based off of known models.*** Models of objects or the environment can help with the estimation of the state of the world around the robot as the robot can recognize known objects and locations. If the models can suggest possible interaction possibilities based on the skills of the robot, action selection can be much improved. Like skills, the object and environment models can either be learned autonomously, pre-programmed, or transferred from another robot.

1.2 Contributions

This dissertation approaches the challenges introduced above in a holistic way by covering all aspects from the robot hardware all the way to software. An interesting robot platform was available in the uBot-5 mobile manipulator [80, 31], but the robot was at its performance limits for many tasks. Therefore, we improved upon its design to remove these limitations and introduced additional capabilities, resulting in uBot-6. It has successfully served as an experimental platform for many years. uBot-7 improves again on this design to create an even more capable platform with more focus on improved manipulation capabilities.

We developed experimental domains for the capabilities of uBot-6. Our control software and planning algorithms were inspired by these domains but were developed to be applicable independently of the robot platform or the experimental domain. The main challenges for planning in these domains were the need for a suitable abstraction to avoid planning at the lowest level and the need to handle partial observability and uncertainty. The need for suitable abstraction is highlighted in the experiments in the mobility domain and has influenced the design of our object models and knowledge representation. Belief-space planning is used to handle the partial observability and uncertainty. As belief-space methods all suffer from poor complexity, the choice of knowledge representation is very important. We use affordance-based object models to provide a good level of abstraction for both of these issues, and we extend them to become more expressive while keeping the complexity low.

The integration of all the required components results in a very capable system with a resourceful robot platform, object models grounded in the robot's capabilities, and a planning framework that can exploit this resourcefulness based on tasks and the environment. The main contributions of this dissertation can be summarized as:

1. ***A unique concept for a robot platform aimed to provide highly versatile capabilities.*** Based on the concept of having 'many solutions for many problems,' the uBot series is a mechanical commitment to solving problems in unstructured environments. It combines many advantageous properties into a single robot platform while remaining easy to use and low-cost. We developed and built two new versions of the series, each with great improvement in capabilities:

   uBot-6 removes the informal barrier between *mobility* and *manipulation*. Usually in robotics, degrees of freedom (DOFs) of wheels are associated exclusively for mobility and DOFs of arms are associated exclusively for manipulation. uBot-6 increases its versatility through action alternatives that use all DOFs for either area. The robot introduces unique access to various postural configurations and the associated forms of mobility.

   uBot-7 extends these capabilities with series elastic actuators (SEAs) to make the robot more robust and safer around humans. It is the first dynamically balancing wheeled robot with SEAs.

For uBot-6, the control framework has been completed, including embedded control, the interface for the ROS middleware, and many controllers. The capabilities of the robots have been modeled empirically, and the robot is used for all of our experiments and demonstrations. The construction of uBot-7 has been completed, but development of all control software is not yet finished.

2. ***A compact but expressive state representation for belief-space planning based on object models.*** We introduce a powerful extension to existing affordance-based object models. The aspect transition graph (ATG) model has been used successfully for belief-space planning [124, 123, 76, 75].

   We extend the ATG model with geometric information to enable several improvements such as easy object pose estimation, more expressive and robust interaction capabilities, robustness to occlusion, and better observation models. As an ATG model is based on perceivable features of an object as well as the interaction possibilities known to the robot, the ATG model can provide abstractions that represent the states of an object that matter to the robot. This abstraction is used to form the state space for belief-space planning. This keeps the state space compact without losing expressiveness. All interaction capabilities are also expressed in an object-centric manner, making them more robust and independent from variance in the robot pose relative to the object. The ATG models provide the state space, transition model, cost estimates, and observation model for belief-space planning. In addition to objects, we also use ATGs to model environments.

3. ***A method to specify generic single-object tasks in belief-space.*** We introduce a method to express different tasks by defining a task partition over the state space and specifying an information-based metric. We use an object-centric state representation, and thus tasks are specified in the context of single

objects. We show how this mechanism supports different task types commonly encountered in robotics.

4. ***A planning framework that incorporates the ATG object models and task specifier to handle single-object tasks in belief-space.*** We present a belief-space planning framework capable of efficient action selection on a real robotic system under partial observability. The framework also handles uncertainty in action outcomes and in observations. State space, transition models, observation models, and empirical cost models are provided by the ATG models. Thus, planning is not slowed down by the need for additional simulation to acquire this information. During planning, the ATG models restrict the search space by efficiently providing only relevant actions to consider. A hierarchical approach supports efficient planning for scenes with multiple objects. We find that finite horizon planning can successfully complete tasks, even for small horizons.

5. ***New planning domains for mobile manipulation on real robots.*** We introduced two new planning domains for belief-space planning that can be used on real robots. These domains are used to evaluate our planning framework in simulation and on the uBot-6 robot.

   The **ARcube domain** is specifically designed to stress planners by providing an extremely high degree of ambiguity between objects in multiple sensor modalities. Sparseness and partial observability of visual features result in a high number of objects that can be visually indistinguishable from a single view point. Additionally, objects can be eccentrically weighted, making some objects only differentiable through their transition dynamics for manipulation actions. Mobility actions can provide new view points around the objects, and manual actions can be used to reorient the objects themselves. It is very easy to create physical copies of the ARcube objects, and they can be adapted to match different size requirements of other robots.

The **dexterous mobility domain** provides a pure mobility domain for robots with multiple forms of mobility. This domain is especially interesting since switching between mobility forms often incurs high costs, which can deter planners from considering these options. A variation of the domain also contains partially observable elements such as doors or blocked passages.

This thesis is organized as follows. In Chapter 2 we introduce background necessary to understand the remainder of this dissertation and discuss relevant existing research. The platform design for the uBot series is presented in Chapter 3 with focus on uBot-6 and uBot-7 that have been developed for this work. It provides an overview of the platforms and their capabilities. In Chapter 4 we introduce our object and environment models, the developed belief-space planners, and a new task representation for belief-space planners. We present a mobile manipulation domain in Chapter 5 that is designed to stress planners through high ambiguity and the need for incorporation of multi-modal information. A variety of experiments is used to evaluate the performance of the belief-space planner. Demonstrations showcase the benefits of the presented task representation and how the they can be used to solve more complex tasks in combination with a higher level decision making. Chapter 6 focuses on dexterous mobility. We define two variants of the dexterous mobility domain with and without partial observability. Exploratory experiments highlight challenges to planners arising from increasingly diverse action costs of resourceful robots. Experiments in the partially observable variant use our belief-space planners. Finally, we provide a discussion of possible future research directions in Chapter 7 and conclude with a summary of the main contributions of this thesis.

CHAPTER 2
BACKGROUND AND RELATED LITERATURE

This chapter provides an introduction to dexterity and its relation to robots and their ability to select actions that exploit perceptual and motor resourcefulness. Section 2.2 discusses related work in mechanism design of mobile manipulators including the background of the uBot series. Section 2.3 describes related work for planning in the context of the dexterous mobility of uBot-6 and uBot-7. Finally, the chapter concludes with a brief overview of belief-space planning and a summary of existing work in that field in Section 2.4.

2.1 “Dexterity”

In daily life, humans (and other animals) encounter many different challenges that they overcome with ease without much thinking. These problems can be known instances that they solved many times before or completely new variations.

An easy to imagine example is the task of opening a door. Effortlessly, we open a large variety of doors every day. They vary in geometry, in the way they open, and how they are unlocked. We adjust to these variations by choosing the more suitable hand, by adapting our grip, or by using more elaborate strategies. In addition to the variations in the door geometry and mechanism, the task can also be constrained by other circumstances, e.g. if we are carrying things in one or both hands. Without much thinking, we use the other hand if our preferred hand is occupied. We might also transfer the carried object to the other hand to free the more suitable hand. If both hands are occupied (e.g. carrying a grocery bag), we might try to open the door with our elbow or, if that seems unlikely to succeed, decide to put the bag down, open the door, and pick the bag up again.

The set of flexible motor skills and rational decision making regarding the use of these skills are collectively called *dexterity.* In the dictionary, one can find definitions such as "the ability to use your hands skillfully," "the ability to easily move in a way that is graceful," and "clever skill: the ability to think and act quickly and cleverly" [55]. Together, these definitions apply *dexterity* generally to body and mind. Nonetheless, in common use, *dexterity* is mainly just associated with fine motor skills with hands.

Similarly, in robotics, *dexterity* is also mostly used in a very narrow meaning: dexterity is generally used only for manipulation tasks and for the structure and control of hands [95]. While Sturges defines dexterity generally as "skill in use with hands" [138], Li *et al.* and Bicchi refer to the ability to change grasps on objects and change the position and orientation of an object in the reach of the hand [89, 14]. Okamura *et al.* make dexterous manipulation object-centric but still dependent on hands, and knowledge of object geometry is even a requirement for dexterous manipulation [101]. Ma and Dollar still treat dexterity only with respect to manipulation and hands but point out that "it's not unreasonable to apply these general definitions to locomotion" [95].

In order to come back to a more general definition that can be applied in robotics, we can look at the definition Nikolai Bernstein coined in the 1940s. In his seminal work "On Dexterity and Its Development," *dexterity* is described as "the ability to solve a motor problem correctly, quickly, rationally, and resourcefully" [13]. This definition expands dexterity to any motor problems instead of just manual skills, and the term *dexterity* can be applied equally to mobility as to manipulation. Dexterity in the mobility domain has been explored by Kuindersma *et al.* and Ruiken *et al.* and is referred to as *dexterous mobility* [80, 116].

Bernstein also notes that the agent must have sufficient fitness and coordination to exhibit dexterity—trying to exhibit dexterity without these features "is like trying to write with a broken pencil" [13]. While resourcefulness in animals arises naturally from evolution, in robotics it needs to be considered specifically by the designer. New robotic platforms must be engineered to have the "sufficient fitness and coordination" necessary for dexterous behavior.

We conclude that dexterity must be the product of the entire design—it does not simply emerge from the mechanism or from control alone. A mechanism design has to provide the physical resourcefulness, but a matching action selection needs to be able to exploit the versatility of the robot. In general, the combination of skills and rational decision making about their use is missing in state-of-the-art robotics and is critical to the successful application of robots in unstructured domains.

2.2 Design for Resourceful Robots

Mobile manipulators are increasingly expected to work in human environments [107, 97, 52, 135, 32]. Research in this area is often conducted with statically stable wheeled robots, such as PR2 and ARMAR-III, that have upper bodies that are equipped with arms and grippers to interact with the environment [16, 4]. These robots can access most human environments that are wheelchair accessible, but they rely on heavy bases with large footprints to ensure static stability even when reaching for objects far away from the body and when maneuvering at moderate to high speeds.

Dynamically stable robots constantly balance to maintain stability, but produce a much smaller footprint and support easier movement in cluttered environments and crowds. Legged humanoids, such as Asimo and ARMAR-4, fall into this category even though they can be statically stable at times [121, 5].

To avoid the complexity of legs on the mechanism in design and control, robots such as the uBot series, Ballbot, and Golem Krang balance on two wheels or a ball to maintain dynamic stability [33, 80, 115, 86, 136]. The use of wheels is also much more energetically efficient than the use of legs. Additionally, dynamic stability can improve performance in tasks like pushing and lifting as the body mass can be used more effectively [33, 70, 79]. This, however, comes at the cost of having to prevent or control possible falls.

The uBot series has been developed as a unique design point combining many advantageous properties into a single robot platform: the uBots have been designed to be large and strong enough to manipulate objects in the real world, yet be small and lightweight enough to be operated easily without complex safety harnesses. At toddler size, the arms can reach table-tops as well as the ground. The small size also makes the robot less intimidating and keeps cost low. Dynamic balancing on two wheels is used to combine great energy efficiency and a small footprint with reduced mechanical and control complexity. Balancing also provides low input impedance longitudinally, making it a good design consideration for safety. Stiff platform impedance laterally supports high manual precision. Its design avoids highly expensive components, such as harmonic drive gears, in order to keep the robot low cost (for a robot with comparable capabilities) while achieving good performance for manipulation and locomotion. The series has been used successfully in many applications ranging from object recognition/manipulation [118], autonomous assembly [117, 139], emergency response [58], to physical rehabilitation [56, 57].

In robotics, mobility is typically just a means to extend the workspace of the manipulator. This usually limits the use of wheel motors to mobility tasks and arm motors to manipulation tasks. We removed this separation between mobility and manipulation resources for the design of the uBot platforms, and the new capabilities result in added physical resourcefulnes, e.g. increased pushing capacity and alternate forms of mobility [140, 80]. This concept is not unique to the uBot series, but it is very uncommon. For example, iCub uses its arm for manipulation and crawling

[34], and NASA Athlete can use its legs for manipulation tasks [147]. The uBot-6 presented in this work (Section 3.1.1) breaks this separation down even further with added postural configurations [115]. Each configuration supports different properties to both manipulation and mobility. Transitioning to a statically stable configuration can greatly increase manual precision and enable successful completion of otherwise impossible tasks. Likewise, different postural configurations also support different forms of mobility that can help overcome obstacles in the environment [116].

The most recent version of the uBot series, uBot-7, is presented in Section 3.1.2. Its design is aimed to improve manual and perceptual versatility by adding series elastic actuators (SEAs) and an improved head [114].

2.3 Planning

Planning on a robotic system can happen on many different levels. This section will cover planning algorithms that are relevant to planning suitable for the various forms of mobility of uBot-6 and uBot-7 (see Chapter 3). First, we provide an overview of related work on path and motion planning suitable to find a path or trajectory for navigation of a mobile robot, then we cover existing work in integrated task and motion planning.

2.3.1 Motion Planning

A navigation task of a mobile robot requires searching for a path in the environment using a set of actions with associated costs. Possibly the most well-known search algorithm is A* [47], which finds an optimal path by combining actual and heuristic cost. While A* is very efficient, the size of the search space as well as time limitations often make it infeasible to find a solution. To address this problem, many anytime variations such as ARA* have been developed [92]. They are able to quickly find a suboptimal solution and then use the remaining time to refine it. Another limitation of A* in real world settings is its inability to efficiently adjust to changes in dynamic environments. Efficient incremental variants of A* such as D* [134] and D* lite [68] are able to repair existing solutions for a fraction of the cost of a complete replan. Both anytime and repair capabilities have been combined in algorithms like AD* [91]. A* type searches have been combined with a state lattice to handle path planning for non-holonomic navigation in state-of-the-art systems [105].

Hierarchical A* (HPA*) uses the basic idea of A* and combines it with a hierarchical search [17]. The search space is divided regularly into larger areas. Transition points along the boundary of neighboring areas are identified, and inside each area A* is used to find shortest paths between all transition points of the area. Using an abstract connectivity graph from these shortest paths, a global path can be found on the higher level. By limiting A* to local searches, the needed resources are reduced, but optimality guarantees are lost. Additionally, the division of the search into regular areas is not applicable to all search space.

The performance of A* type searches is highly dependent on the quality of heuristics. Sometimes it can be difficult to find a heuristic that captures the properties of the system well enough to efficiently guide the search [48]. A major problem is heuristic depression regions in the search space where the heuristic values do not correlate well with actual cost-to-goal. This results in an exhaustive exploration of less expensive actions before necessary higher cost actions are considered. A possible solution is the usage of distance-to-go as part of the search heuristic instead of or together with the cost-to-go [119, 148, 149]. The number of steps to the goal replaces the estimated cost to the goal. This is equivalent to unit costs for all actions. A decision has to be made ahead of time what trade-off between speed and quality is desired.

Recently, the use of sampling-based algorithms, such as Rapidly-exploring Random Trees (RRT) [87, 88] and RRT Connect [78], have become more popular for path planning in navigation [20] as well as motion planning in configuration space [77]. These algorithms tend to find a solution very quickly, but give no guarantee for optimality. They struggle for problems in which a solution path has to pass through a very constrained region. While they are probabilistically optimal, finding a solution in such cases would take too much time. Variants such as Multipartite RRT [155] provide support for dynamic environments.

Path planning using Harmonic Functions (HF) [26, 27] offers navigation similar to using potential functions [65] without local minima. The resulting paths are smooth and HFs handle changes in the environment well. The major limitation for HF was that gradients would get too shallow for large map sizes for long paths with bottlenecks. These limitations were removed recently by expressing HFs in log-space [154]. Nonetheless, it is not straight-forward how to use HF for planning non-holonomic paths.

Probabilistic roadmaps (PRM) can be built by sampling configurations and connecting them with "reachable" neighbors using a simple but fast local planner [64, 63]. Edges are introduced if a path exists and labeled with the corresponding cost. Usually PRMs are constructed in a preprocessing step, but query and construction steps can be interleaved as construction is incremental.

For most of these methods, it is not straightforward how to use them for planning non-holonomic paths. This makes the heuristic search with an A* type method in combination with a state lattice the most promising approach for planning paths for the prone-scooting mobility form of uBot-6 and will be discussed further in Section 6.2.3.

2.3.2 Task and Motion Planning

In this section we discuss some related works that combine task planning and motion planning into unified frameworks as they can be relevant to the approach used for hierarchical abstraction for dexterous mobility in Chapter 6.

DARRT [9] is a planning algorithm that uses rapidly exploring random trees (RRT) [87, 88] to find plans for manipulating objects. The search is performed in the combined configuration space of the robot and object. The result is a configuration space trajectory for the whole task. Sampling based algorithms are generally suitable for motion planning in configuration space, but have special requirements to deal with non-prehensile manipulation, as for example an object cannot move by itself, and the trajectory is tied to the arm trajectory during a pushing action. To overcome this problem, DARRT uses projection functions to guide samples towards interesting configurations based on manipulation primitives and on object trajectory.

The faster version, DARRTHConnect [10, 11], uses the bidirectional variant DARRTConnect [10] in a hierarchical manner. A relaxed path for the object is calculated, and subgoals are extracted from this path. DARRTConnect is used to solve each subtask. The separation into subtasks results in faster planning times as only part of the plan has to be redone if DARRTConnect gets stuck and needs to be restarted.

DARRT quickly finds solutions to interesting manipulation problems that might include non-prehensile interactions and tool use. But the use of these planners is limited as the result of the planning process is an open-loop trajectory in configuration space with the associated problem: generally, long open-loop plans have increasing failure rates with increasing lengths as positioning errors accumulate. Open-loop movements are also not tolerant to faults as the robot is oblivious to when parts of the trajectory did not fulfill its intended purpose (e.g. a hand closed next to an object, but the robot continues as if the object was grasped successfully). Additionally, these methods require full knowledge of the environment, which is unrealistic in most situations.

Kaelbling and Lozano-Pérez presented the hierarchical planning in the now framework (HPN), which is a combined framework for task and motion planning which plans “in the now” [60, 61]. The goal and intermediate planning states are represented symbolically. An abstract plan is found for the given conditions recursively. Preconditions are dropped for abstract actions to simplify planning. An abstract action can be resolved into new “sub”-conditions and the recursion is repeated. Once a primitive action is reached during planning, it is executed. Only the earliest conditions and subtasks are expanded resulting in only the part of the plan being refined that needs to be executed first. As a result, the planner can deal with large horizon problems. The planner itself is a goal regression planner which starts from the goal and searches towards the start. Instead of enumerating all precursors, geometric “suggesters” are used to provide some sample poses and locations. Search on the highest level is done by A* with the number of unsatisfied conditions as the heuristic.

This approach aggressively sacrifices optimality in favor of efficiency. The success of this approach rises and falls with the quality of the geometric “suggesters.” While this work still uses full knowledge about the environment, the planning “in the now” approach seems very appropriate as future and unknown situations are likely to be unknown or different in partial observability. Like the DARRTH framework, the final outcome of this framework are open-loop trajectories as result of each primitive action. As they are calculated only once the execution reaches that part of the plan, it suffers less from accumulated errors.

RCHPN is an extension to HPN that reorders and combines subtasks to reduce execution cost [46, 45]. It is still not using cost estimates for abstract actions, but orderings that would include subtasks, that need to be undone later, can be avoided.

Wolfe *et al.* [151] provided a framework for integrated task and motion planning based on Hierarchical Task Networks (HTNs) [99]. Dornhege *et al.* proposed the usage of semantic attachments, which enables a low level planner to change fluents in the description language, such as PDDL, used by a high-level planner at runtime [38].

Stilman *et al.* present a planning framework for the problem of Navigation Among Movable Obstacles (NAMO) [135]. A robot needs to find a path through an environment with fixed and movable obstacles. The robot needs to use its manipulators to move the obstacles out of the way to create a path to the goal. The task is split into several smaller path planning problems and manipulation tasks. Then the algorithm finds which combination of subproblems needs to be solved to find a solution to the complete problem at minimal execution cost. The result are open-loop mobility and manipulation trajectories, but could also be returned as a list of more abstract subgoals. A probabilistic extension can solve more difficult problems, but suffers from possibly large number of unnecessary manipulation actions [144]. This work has also been extended to manipulation planning [137].

With respect to dexterous mobility, all these approaches use some level of abstraction and could be used to find solutions similar to those from the approach used in Section 6.2.4, but their success is likely mostly dependent on the quality of the generated subgoals.

2.4 Belief-Space Planning

Robots usually encounter a lot of uncertainty in observations and action outcomes. Observations can be uncertain because of factors like sensor noise and occlusion. Variances in the robot's controllers and state estimation can cause non-deterministic action outcomes. Additionally, the state of the robot and the environment is usually only partially observable. In structured environments, the impact of these issues can often be controlled such that they do not have to be handled explicitly. In unstructured environments, a robot usually needs to be able to operate despite these issues. A typical method to handle partial observability and uncertainty is to perform planning in belief space. In order to make decisions under uncertainty in the state, the robot maintains a belief in the form of a probability distribution of the true state. The robot can take actions and observations to update this belief. There are many different approaches for belief-space planning that sometimes share methods and also problems. We will discuss some of these approaches that are relevant to this dissertation.

2.4.1 Forward Model

A necessary component of a belief-space planner is a means of propagating belief distributions through candidate actions using a forward model. For example, Hogman *et al.* use the action-effect relation to categorize and classify objects [51]. Loeb and Fishel discuss how Bayesian Exploration can be used to construct queries to associative memory structures of previous sensorimotor experiences [93]. Browatzki *et al.* use a similar action selection metric and transition probabilities on a view sphere with a set of actions that execute in-hand rotations [19]. Wörgötter *et al.* propose the affordance-based object-action-complexes (OACs) to model the transition model of objects for different actions [153]. Sen introduces similar affordance-based object models called aspect transition graphs (ATGs) that combine bag-of-features feature matching with a graph to model action effects [125, 122]. We extend the ATGs by adding geometric information and cost estimates to improve state representation and forward modeling capabilities in Chapter 4.

2.4.2 Active Perception

Belief-space planning is intuitively well suited for use in the field of active perception as actions need to be chosen to reduce uncertainty. There is substantial evidence that human visual learning is facilitated by coupling perception and action [42]. The active vision community advocates exploiting actions that change sensor geometries in order to actively improve confidence in perceptual feedback and information gain [7]. Aloimonos *et al.* introduce the first general framework for active vision in order to improve the quality of tracking results [2]. More recently, Denzler *et al.* demonstrate how altering sensor geometry can be used effectively to deal with limited field-of-view and occlusions in the scene [35]. Sen *et al.* model the interaction capabilities of a robot with an object in affordance-based object models [126, 125]. They show how distributions over populations of object models can be used for greedy action selection to reduce entropy over object models [122, 124]. Sen and Grupen showed that by pruning models with insufficient support, it can scale to large numbers of models (up to $10,000$) [123].

This thesis uses a generalization of seminal contributions from the active vision community [7, 2] that can be applied to multi-modal perceptual information and to general-purpose problem solving. We present an active belief planning framework that builds on the work of Sen in Chapter 4.

2.4.3 Task Switching

Often the robotics community works on when to switch between tasks [146, 21] rather than how to solve different active perception tasks using a single planner. Grabner *et al.* proposed a single framework to solve both object identification and object categorization in object recognition problems [44]. Lai *et al.* proposed a scalable tree-based approach to solve category recognition, instance recognition, and pose estimation [84]. These methods, however, are not active recognition algorithms, and therefore, they do not interact with the environment to reduce the uncertainty. As discussed in Chapter 4, we combine active perception with a mechanism to seamlessly switch between tasks [117].

2.4.4 Partially Observable Markov Decision Processes

A popular formalism for reasoning in a system with uncertain action outcomes is the Markov decision process (MDP) framework [12]. If the state of the system is not fully known and only partially observable, then the system can be formalized as a partially observable Markov decision process (POMDP) [131, 22, 59]. There exist many different algorithms to solve POMDPs, however, their usage is limited as POMDPs quickly become computationally intractable with a growing number of states, observations, and actions.

POMDPs have been used successfully for tasks such as robot navigation [113] and robot interaction [30]. The size of the state space required can still be prohibitive, however. To scale these approaches, Castanon uses a hierarchical POMDP to recognize many objects in a scene [24], and Sridharan *et al.* introduce a hierarchical POMDP to plan visual operators to recognize multiple objects in the scene [133]. Often planning with a small finite horizon can already provide good solutions to POMDPs. Eidenberger and Scharinger formulate an approximate solution for a POMDP with a horizon 1 value function. They demonstrated that this approach generates next viewpoint actions that successfully recognize multiple objects in a cluttered scene [40]. Hsiao *et al.* utilize a decision theoretic solution to a POMDP to determine the relative pose of a known object [53]. They show that rolling out belief states by just two plies can lead to a drastic decrease in the number of actions, but that multi-ply planning quickly becomes computationally prohibitive. Araya *et al.* note that the reward structure of POMDPs can be prohibitive when the distribution of belief itself is critical for the task [3]. For our belief-space planner, we combine finite horizon planning and handle multi-object scenes in a hierarchical manner to support efficient action selection in Sections 4.3.4 and 4.3.2.

Optimal solutions to POMDPs are provided by offline solvers that compute an optimal policy but are generally intractable for real robot problems. Online planners for POMDPs address this problem by planning up to a finite horizon and then choosing the best action at that plan depth [112, 130]. Point-based algorithms offer much increased performance and make it possible to solve much larger POMDP problems [103, 104, 132]. These algorithms generate a set of belief points to approximate the full belief space. This set is much smaller and enables a more efficient computation of an approximate solution only for these belief points. The quality of the solution is dependent on how well the sampled set of belief points approximates the belief space. HSVI1 [128], HSVI2 [129], and SARSOP [83] utilize heuristics to direct the sampling process towards the reachable areas of the belief space in order to improve the approximation with fewer belief points. In order to evaluate the performance of this class of solvers for our planning domains, we combine point-based value iteration (PBVI) with the ATG models as an online solver for POMDPs as shown in Section 4.4 [103].

2.5 Discussion

Operating successfully in unstructured environments is difficult for state-of-the-art robots. The platforms often have specific capabilities and strategies for given tasks. If this strategy cannot be used, often no alternatives are available. With the uBot series, we provide a very resourceful robot and further extend its capabilities with functional equivalent action alternatives for locomotion as shown in Chapter 3.

The robot needs to choose between alternative actions or action sequences that can complete a task. Many different belief-space planning approaches might be suitable. We are interested in a single approach that can handle a large variety of tasks instead of many special purpose solutions. Information-theoretic planners perform well even with finite horizon planning to small depths, and they are well suited for when the distribution of belief itself is critical for the task while POMDP solvers struggle here. On the other hand, for tasks that have specific states as goals, the small search depth could be prohibitive. POMDP solvers might be more suited here, though they have tractability issues with respect to the planning time. We combine a version of each with our ATG object models to evaluate their performance (Chapter 4). We choose PBVI as a representative for the POMDP solvers. In the future, a heuristic based alternative could be very successful in our domains.

CHAPTER 3
A PLATFORM FOR STUDYING "DEXTERITY"

In the seminal work "On Dexterity and Its Development" by Nikolai Bernstein, *dexterity* is described as "the ability to solve a motor problem correctly, quickly, rationally, and resourcefully" [13]. This definition makes it clear that dexterity must be the product of the entire design—it does not simply emerge from the mechanism or from control alone. Often the dexterity of a robotic platform is judged by its degrees of freedom or by the capability of manipulating items in its hand. These are very limiting views on what actually determines the dexterity of a robot. A robot can have a high number of degrees of freedom (DOF) and still be completely incompetent if it does not know how to use those DOFs to solve a given task. A robot can also be very dexterous in solving a single task under many different constraints, but be helpless in many other tasks.

Our focus is on the "resourcefulness" specification. In this case, "dexterity" implies that when a prototype solution cannot be applied due to run-time constraints, a robot can modify its strategy quickly to deal with the unexpected circumstances. Our approach is to design mechanical systems with different types of solutions with varying performance characteristics that can be exploited by matching control and planning approaches when required.

3.1 uBot Mobile Manipulators

Based on the concept of having 'many solutions for many problems,' the uBot series is a mechanical commitment to solving problems in unstructured environments.

It has been developed as a unique design point combining many advantageous properties into a single robot platform: the uBots have been designed to be large and strong enough to manipulate objects in the real world, yet be small and lightweight enough to be operated easily without complex safety harnesses. At toddler size, the arms can reach table-tops as well as the ground. The small size also makes the robot less intimidating and keeps cost low. Dynamic balancing on two wheels is used to combine great energy efficiency and a small footprint with reduced mechanical and control complexity. Balancing also provides low input impedance longitudinally, making it a good design consideration for safety. Stiff platform impedance laterally supports high manual precision. Its design avoids highly expensive components, such as harmonic drive gears, in order to keep the robot low cost (for a robot with comparable capabilities) while achieving good performance for manipulation and locomotion.

As part of this work, two new versions of the uBot series have been developed based on experience with previous versions: uBot-6 and uBot-7 [115, 114]. An overview of their basic specifications compared to their predecessor uBot-5 can be found in Table 3.1.

Table 3.1. Basic specifications of the uBot-5, uBot-6, and uBot-7 robots.

	uBot-5	**uBot-6**	**uBot-7**
Body height (to top of head)	77 cm	88 cm	95 cm
Body width	59 cm	62 cm	65 cm
Body depth	20 cm	23 cm	25 cm
Body mass	18.9 kg	24.8 kg	27.0 kg
Arm length (shoulder to hand)	57 cm	70 cm	74 cm
Base wheel diameter	20.5 cm	23 cm	23 cm
Elbow wheel diameter	—	7 cm	7 cm

In their primary form of locomotion, the robots are dynamic balancers on two wheels. The wheels provide non-holonomic drive capabilities with differential steering. They are capable of transitioning to and between other postural configurations. The postural configurations and corresponding mobility forms are available to both robots and are detailed in Sections 3.2 and 3.3. Each postural configuration enables additional ways to interact with the environment and solve tasks. At the same time they also pose additional requirements to the robot design.

3.1.1 uBot-6

uBot-6 is a toddler-sized mobile manipulator that balances dynamically on two wheels. Experiences from the design and operation of uBot-4 [33, 140] and uBot-5 [31, 80] have been incorporated into the design of uBot-6.

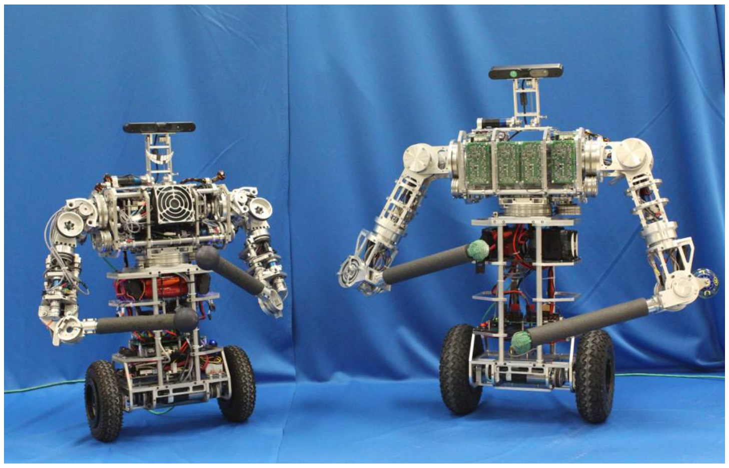


Figure 3.1. Two whole body mobile manipulators with 13 degrees of freedom side by side: uBot-5 retrofit with a uBot-6 head (left) and uBot-6 (right).

The robot has 13 DOFs: two wheels, a rotatable trunk with two 4-DOF arms, and a 2-DOF head. Added in uBot-6, the elbows are equipped with small, unactuated wheels that enable additional forms of mobility (see Section 3.3). The basic physical dimensions of the uBot-6 robot are summarized in Table 3.1. uBot-5 and uBot-6 are shown side by side in Figure 3.1. Compared to uBot-5, weaknesses in joint transmissions have been eliminated and the strength of all joints has been increased least threefold. Additionally, the hands have been equipped with force/torque sensors

3.1.1.1 Head

A new 2-DOF head (Fig. 3.2) has been designed to complement the postural versatility of the robot: two coupled joints driven by one motor result in a combined rotational and translational movement that permits full view of the bimanual workspace in all postural configurations. Examples of the requirements due for the head are shown in Figure 3.3. An RGB-D camera (Asus Xtion Pro Live [6]) provides color and depth information from the manual workspace.

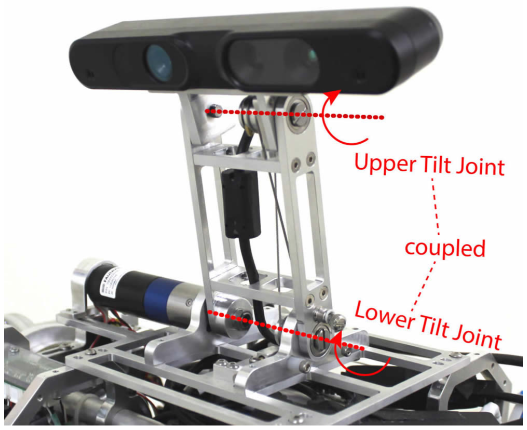


Figure 3.2. The uBot-6 head: One motor drives two coupled degrees of freedom resulting in a combined rotation and forward-backward translation. This mechanism permits full view of the bimanual workspace in all body postures. Panning of the head can only be accomplished through torso rotation while standing upright.

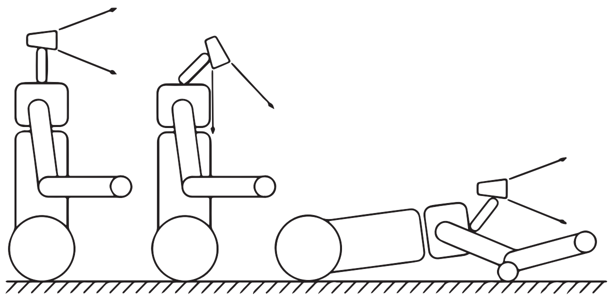


Figure 3.3. Examples of different requirements for camera direction: upright balancing (left), unobstructed view of the manual workspace in front of the robot (middle), four-point contact postural configuration for prone scooting (right).

3.1.1.2 Arms and Torso

The arms, torso, and wheels of uBot-6 are driven by brushed DC motors. Compared to uBot-5, the strength of all joints has been increased at least threefold, and several weaknesses in joint transmissions have been eliminated.

Spherical end effectors are connected to the arms through ATI Mini45 force/torque sensors to provide better manipulation capabilities. They have been chosen such that the measurement resolution is maximized while being robust enough to withstand the impact of a potential fall.

3.1.1.3 Computation Architecture

High performance low-level motor control at 2 kHz is provided by a custom field programmable gate array (FPGA) board. An on-board quad-core computer runs Linux and ROS [109] Indigo to implement mobility and manipulation controls as well as vision processing and behavioral programs. Wireless connectivity allows additional computation to be performed on external computers as well as communication between several robots.

3.1.2 uBot-7

With advances in mobile manipulation, interest is increasing in robots working side by side with humans. Safety is the main concern in such scenarios. Accidental impacts with humans should have minimal potential of injuring them and need to be detected by the robot. The most popular method to implement this functionality in robots is the use of impedance control [50]. This can either be done on robots with series elastic actuators (SEAs) [108] with passive compliance at the expense of precision, for example Baxter and COMAN [110, 143], or on robots with high performance torque sensing capabilities, for example DLR's Justin [102, 150].

uBot-7 (Fig. 3.4) has been developed to incorporate SEAs into a versatile platform such as predecessor uBot-6 [114, 28, 29]. The toddler-sized mobile manipulator now has 14 DOFs. The robot weighs about 27 kg and is 95 cm tall. Like uBot-6, it has two 4-DOF arms and a rotatable trunk. The head has been improved and has three independent, active degrees of freedom. Extended details can be found in focused publications on: the mechanical design of robot with sensor and motor selection [28], the modular SEA modules [29], and the complete design concept of uBot-7 including the head design, embedded electronics, and compute architecture [114].

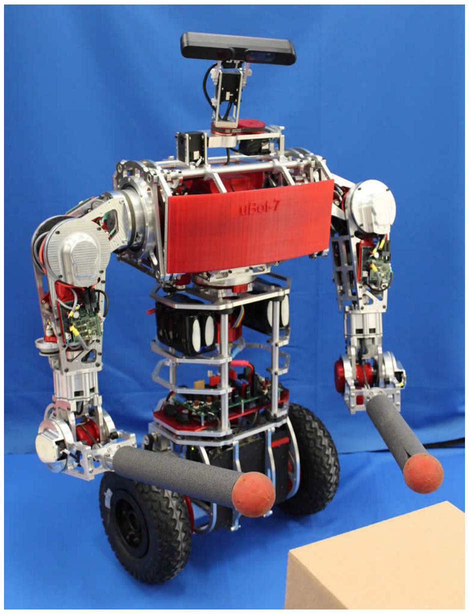


Figure 3.4. The uBot-7 mobile manipulator: a dynamically stable robot with 14 active degrees of freedom and 10 series-elastic joints. It weighs 27 kg and is 95 cm tall.

3.1.2.1 Head

The new improved head mechanism extends the benefits provided by uBot-6's head. The head of uBot-6 uses two coupled joints driven by a single actuator to enable compensation of different body angles due to dynamic balancing as well as different postural configurations (Figure 3.3), but panning of the camera can only be achieved by rotating the torso and is only available while balancing upright. Additionally, it enables translating the RGB-D camera forward to provide an unobstructed view of the space in front of the robot.

The head for uBot-7 keeps these capabilities, but enables head panning and tilting independently of the torso rotation joint. It uses three active degrees of freedom: a tilt joint at the base followed by a pan and a tilt joint on top (Figure 3.5). The lower tilt joint matches the capabilities introduced in uBot-6. The pan-tilt unit on top adds the much needed capability to point the RGB-D camera (Asus Xtion Pro Live [6]) independently of the postural configuration and body angle. All head joints are based on Dynamixel MX-28 servos with integrated sensors. Joint characteristics can be found in Table 3.2.

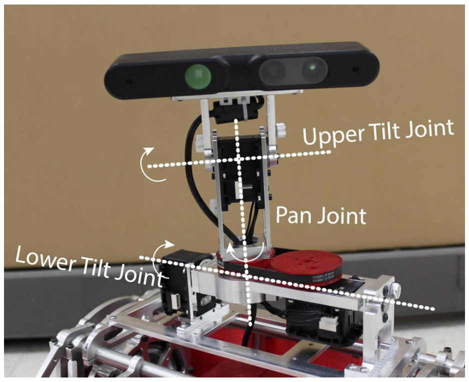


Figure 3.5. The uBot-7 head with three active degrees of freedom: tilt, pan, tilt. The first tilt joint supports compensation of the body angle of the dynamic balancer and enables pan-tilt motion in all postural configurations. Axes of rotation for each joint are shown in white.

3.1.2.2 Arms and Torso

In uBot-7, each arm and torso joint is driven by an SEA. The addition of SEAs to all arms and torso joints extends the passive, anisotropic impedance character of its base into an active impedance character in the upper body. The SEAs support impedance control for safer operation near humans and also provide force/torque

Table 3.2. Joint characteristics for uBot-6 and uBot-7.

		Cont. torque [Nm]		**Max. speed** [deg/s]		**Range of Motion** [deg]
		uBot-6	uBot-7	uBot-6	uBot-7	uBot-7
Head	**Upper Tilt**	–	0.6	–	400	[-90, 90]
	Pan	–	0.6	–	400	[-200, 200]
	Lower Tilt	0.6	0.6	850	400	[-90, 90]
Torso		2.6	12.0	570	234	[-165, 165]
Shoulder	**Roll**	15.0	45.0	100	114	[-360, 360]
	Pitch	6.4	21.0	237	162	[-30, 180]
	Yaw	6.4	12.0	237	234	[-80, 260]
Elbow		6.4	12.0	237	234	[-60, 100]
Wheel		2.4	4.2	900	1900	continuous

sensing for better manipulation capabilities. To our knowledge this makes it the first wheeled dynamic balancer with SEAs.

In order to incorporate SEAs in several different joints, we developed a modular package that can be customized to the various requirements of different joints [29, 28]. We combine flat brushless DC (BLDC) motors, planetary gearheads, and flat torsional spring designs in order to create a module that is lightweight, low-volume, easy to reconfigure, and high performing. The spring design has a linear torque-displacement relationship. It was developed in collaboration with partners at the Johnson Space Center and is similar to the spring design used in NASA's Robonaut 2 [36, 54].

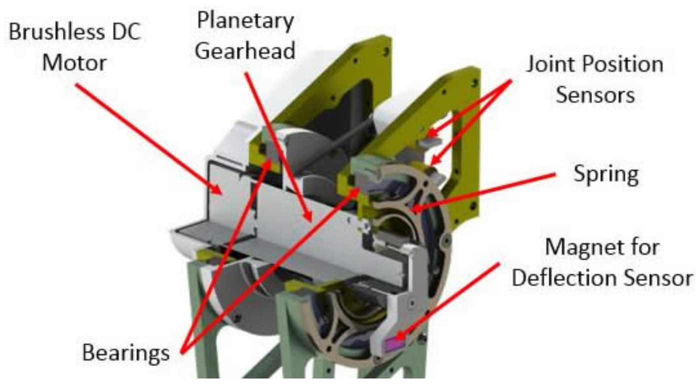


Figure 3.6. Components of the SEA for shoulder abduction.

Absolute position sensors measure the position of the joint output at high resolution. Joint torque is determined by measuring the deflection of the torsional spring directly with a Hall effect sensor.

All components of the SEA module (except the springs) are commercially available and affordable enough to match the low-cost principle of the uBot series. The spring design is based on two-dimensional cutting operations and can be machined easily.

The experience with uBot-5 and uBot-6 was used to select appropriate joint torques, velocities, and ranges of motion (Table 3.2). As the uBot can balance on a differentially steered, two-wheeled base and may fall, the resulting loads on the arms may exceed normal design loads. The passive properties of SEAs combined with reactive control can protect gearheads from damage. We chose a maximum spring deflection of $\pm 4^\circ$ as a trade-off between passive compliance and torque sensing resolution. The spring constants for each joint were chosen based on maximum joint torque and the maximum spring deflection. This results in sensing resolutions between 0.011 Nm (elbow) and 0.044 Nm (shoulder). The spring constant of the used springs is linear with respect to the thickness of the spring. Thus, the spring constants can easily be changed by installing thicker or thinner springs.

3.1.2.3 Drive System

The main drawback of the uBot-6 drive system has been the presence of backlash in the gearheads of the drive motors. As balancing requires frequent direction changes just at the equilibrium point, this backlash can be very noticeable. With every direction change, the gear first moves through the backlash region before providing torque in the other direction. As a result, wheel chattering can occur when standing still and only small wheel corrections are required. This issue has been addressed in a new drive train design for uBot-7. Powerful 90 W flat BLDC motors provide much higher torque to the drive wheels and thus only a very small additional gear reduction is needed. We use timing belts to provide a backlash-free 7.5:1 gear reduction to the wheels. The change roughly doubles available continuous torque, maximum torque, and maximum velocity while removing backlash issues at the same time. A specifically designed embedded control board handles control of both wheel motors as well as measurements from inertial measurement units (IMUs) to ensure all resources required for balancing are in one place for improved reliability. The control computer calculates shifts in the center of mass of the robot based on all joint angles and passes them as parameters to the balancer running on the embedded hardware.

3.1.2.4 Computation Architecture

uBot-6 employs a centralized motor control strategy and controls all joints from a single field programmable gate array (FPGA). On uBot-7, motor controllers are distributed over the whole robot and placed close to the respective joints. Sensors and motors are interfaced locally, and only a communication bus and power need to be routed. As a result only a small number of cables run through the arms, simplifying routing. The freed space in the torso can house additional on-board computing hardware.

Custom embedded controller boards for motor control and sensor processing are distributed throughout the body in proximity to the respective arm, toro, and wheel joints. The controller boards are built around Maxon ESCON 50/5 motor control modules. A PIC32MX 80 MHz microcontroller interfaces the motor control modules, sensors, and the communication bus. It currently supports position, velocity, torque, and impedance control at 1 kHz control rate and allows full access to extend the programming.

Two different custom board versions are employed in the robot: for a single motor (Fig. 3.7 (left)) and for two motors (Fig. 3.7 (right)). All arm joints as well as the torso rotation are each equipped with a single embedded controller board per joint. The wheel motors share a control board.

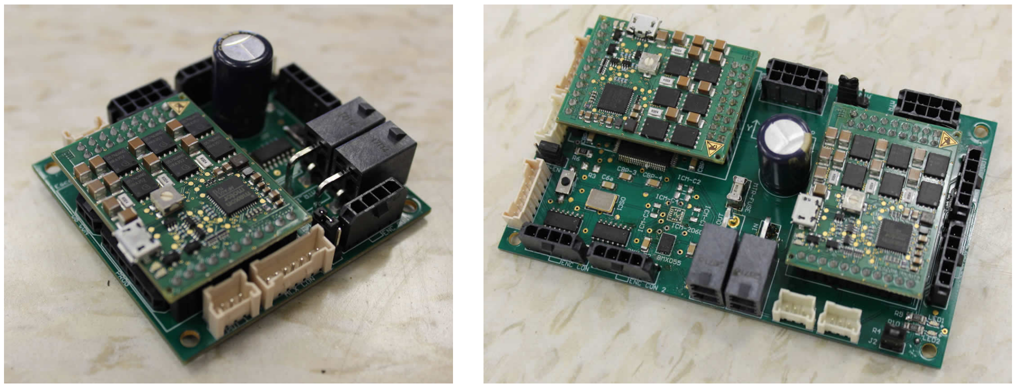


Figure 3.7. Custom embedded controller boards for motor control and sensor processing for a single motor (left) and two motors (right). They support motor encoders, several absolute position sensors for measuring joint output positions, torque sensing, and inertial measurement units (IMUs).

The robot is equipped with two on-board computers. An embedded pico-ITX computer with a quad-core 6^{th} generation 2.6 GHz Intel Core-i7 processor is housed in the base of the robot. It handles communication with the distributed motor controller and sensor boards and any Cartesian control. Additionally, it can perform high-level planning and control. A Jetson TX1 board adds capabilities for GPU accelerated processing for applications like vision, planning, and deep learning [100, 154]. Additional space is available to add more computer hardware to the robot in the future.

An overview of the computer and communication architecture used on uBot-7 is shown in Figure 3.8. The on-board computers communicate over Gigabit Ethernet. A mini-router provides reliable WiFi connectivity for all on-board computers with the ability to secure all off-board communication through a virtual private network (VPN).

The distributed motor controllers for wheels, arms, and torso are connected to the control computer through a RS-485 communication bus running at 10 Mbps which is fast enough to support real-time control. The head motors are connected through another RS-485 bus running at 2 Mbps.

A software interface exists for the robot operating system (ROS) that supports interfacing to all joints and sensors of the robot for easy application development [109]. The interface is compatible with previous uBot versions and existing applications will transfer to the new robot with minimal changes.

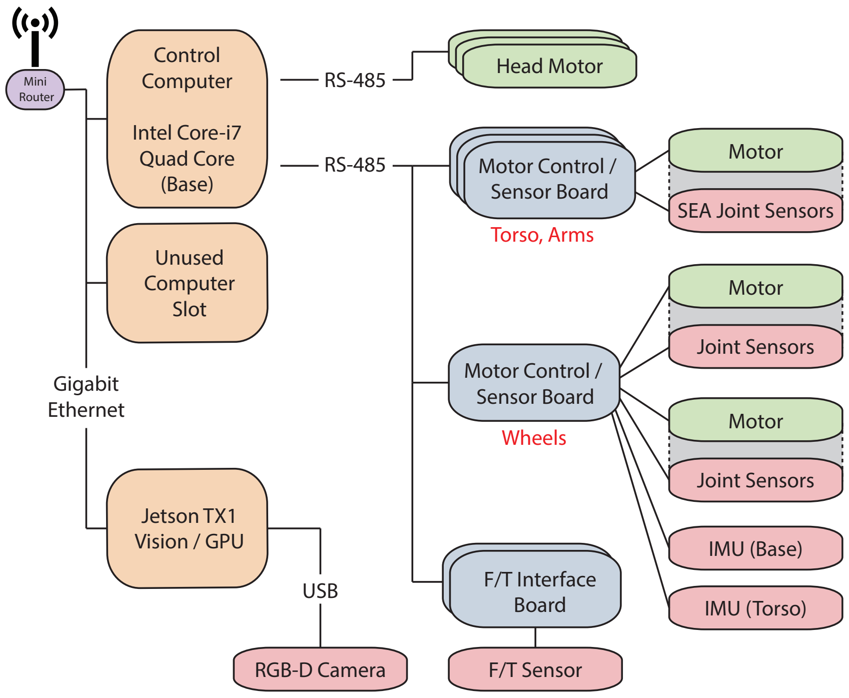


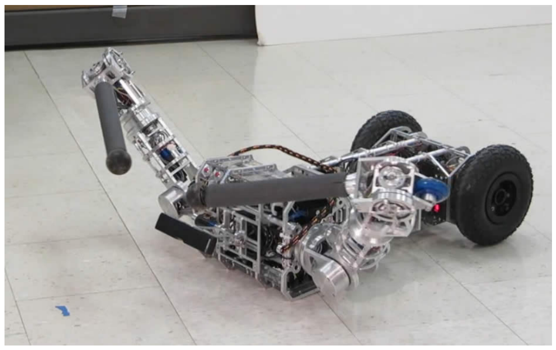
Figure 3.8. Computer and communication architecture of uBot-7.

3.2 Postural Configurations and Transitions

uBot-6 and uBot-7 are designed for whole body dexterous mobility and manipulation. They can assume several postural configurations (see Fig. 3.14). Each postural configuration has different characteristics that enable or constrain manipulation and mobility capabilities.

3.2.1 Prone

In this configuration the robot lies on the anterior surface of its body. The trunk rotation is disabled and both arms are free with a limited overall reachable workspace, a significant portion of which is in the ground plane. The robot footprint is relatively large in this configuration and the body height is minimized.


(a) uBot-6

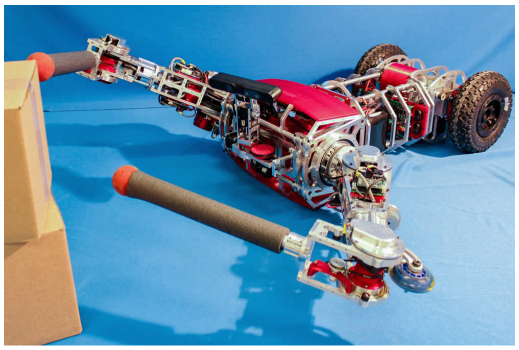

(b) uBot-7

Figure 3.9. Prone postural configuration: Lying prone provides maximal stability and leaves the arms free for manipulation.

3.2.2 4-Point Stance

In this stance, the base wheels as well as both arms are in contact with the ground. The arm-ground contact is established with hands and/or elbow wheels (Fig. 3.10 and 3.11). In case of ground contact with the elbow wheels, the hands can still be moved in the nullspace and used for manipulation actions. For uBot-6, this posture allows the body height to vary between 24 cm and 88 cm.

3.2.3 3-Point Stance

This stance is a quasi-statically stable tripod with both base wheels and one arm in contact with the ground (Fig. 3.12). The arm-ground contact is established either with a hand or an elbow wheel. There are an infinite number of ways to achieve this configuration by varying the choice of left or right arm, hand or elbow ground

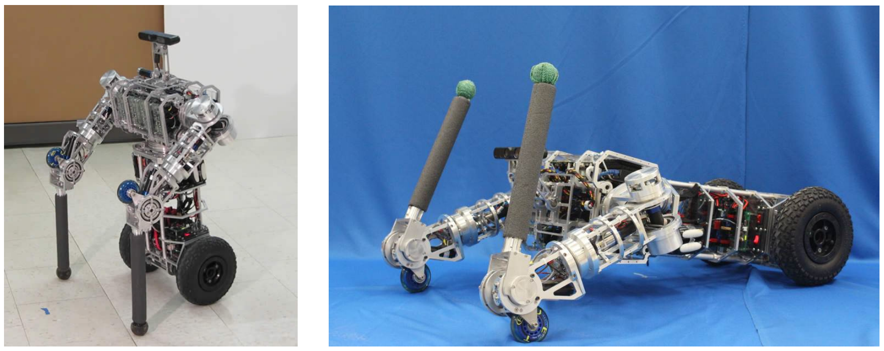

(a) (b)

Figure 3.10. 4-point stance using hands (left) or elbow wheels (right).

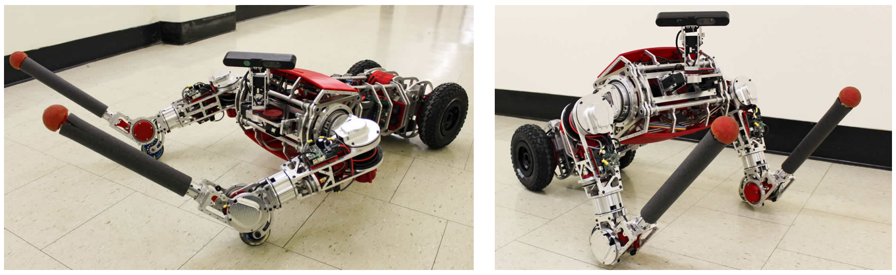


Figure 3.11. Examples of two statically stable postural configurations of uBot-7: 4-point contact with elbow and base wheels at low body height (left) and high body height (right) support alternative forms of mobility with reduced manipulation capabilities.

contact, body height, and location of the ground contact. The choice of body height and ground contact location are constrained by the stability requirement of keeping the zero moment point (ZMP) inside the triangle formed by the three ground contacts [145, 80].

The second arm is free for manipulation actions and only constrained by the stability requirement. Therefore, only movements that do not shift the ZMP out of the support triangle are allowed. As in the case of the 4-point stance, the body height for uBot-6 can be varied between 24 cm and 88 cm.

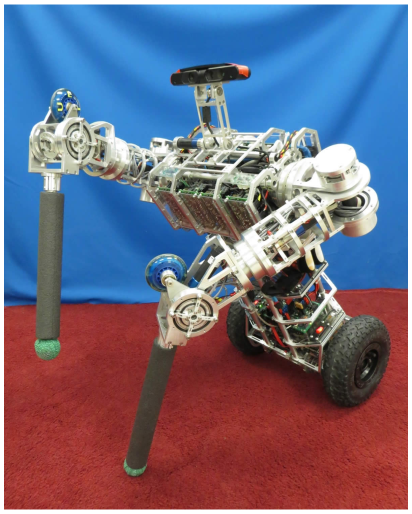


Figure 3.12. A 3-point postural configuration of uBot-6.

This postural configuration is used in 5.4.3 to acquire better visual observations. Once a hand touches the ground, the robot is statically stable and locations associated with visual features become more precise.

3.2.4 2-Point Stance

The robot stands upright by balancing on its two base wheels (see Fig. 3.13 and 3.4). A Linear Quadratic Regulator (LQR) runs on the on-board FPGA and controls the balancing at 1 kHz. In this postural configuration, the robot has a minimal ground footprint. The dynamic stability of this posture results in low longitudinal input impedance. This contributes to safety in the presence of humans.

In this posture, the arms of the robot are free to perform manipulation tasks. Trunk rotation and the symmetric workspace of the arms contribute to a large bi-manual workspace. This includes a large area on the ground plane while providing accessibility to counter tops and low shelves. The robots use a forward compensator to anticipate the effect of their own upper body posture on the center of mass in order to improve the end point precision. A damping term suppresses dynamic effects of rapid arm movements.

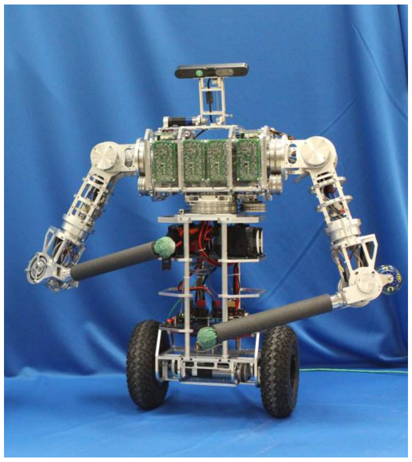


Figure 3.13. 2-point stance of uBot-6 allowing for energy efficient balancing mobility with differential steering.

3.2.5 Postural Transitions

The robot can reliably transition between the different postural configurations by following a path through the postural transition graph as depicted in Figure 3.14. Additional transitions such as the transition from a 4-point-hand stance to a 4-point-elbow stance without passing through prone configuration can easily be imagined. Implementation details for these controllers are presented by Kuindersma *et al.* [80].

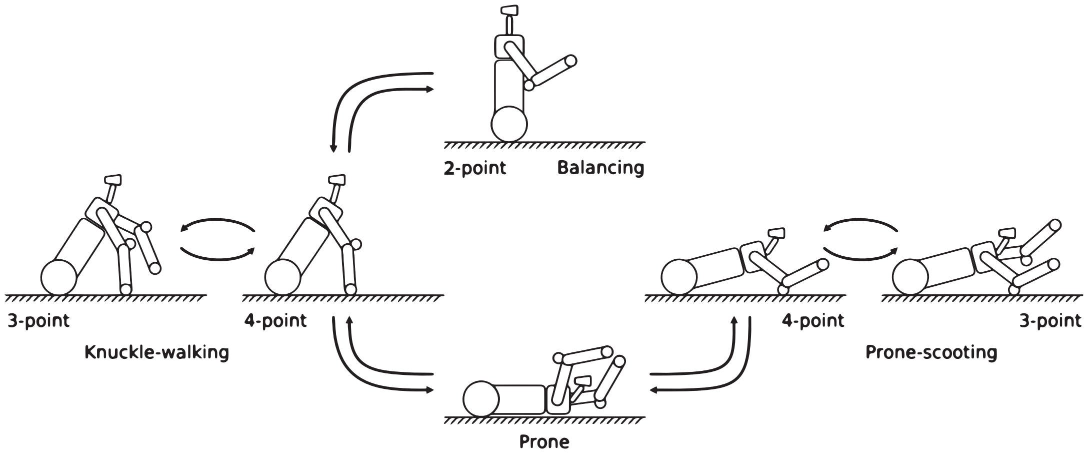


Figure 3.14. Postural transition graph for uBot-6 and uBot-7. Postural options include: 2-point contact during dynamic balancing, 3 and 4-point contact during knuckle-walking and prone-scooting, and prone resting.

A transition can only be executed if the desired postural configuration is compatible with the current location and if sufficient space for the transition is available. Time and energy costs for common postural transitions are shown in Section 3.3.4.

3.3 Mobility Capabilities

uBot-6 and uBot-7 have multiple forms of mobility at their disposal. Each mobility option offers unique possibilities to move in various environments. In this section, we highlight and characterize three of these forms of mobility: balancing mobility, prone-scooting, and knuckle-walking. Other possible forms of mobility such as elbow-walking are left for future work. Each form of mobility uses one or more postural configurations for support.

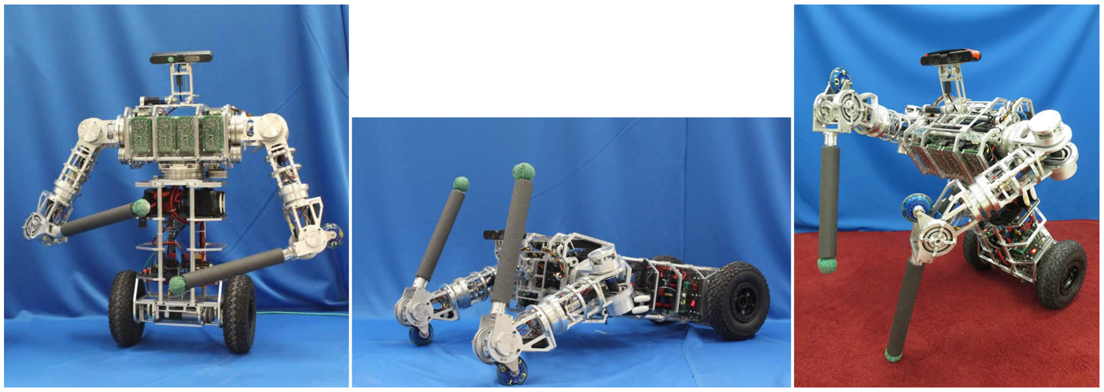


Figure 3.15. uBot-6 in several postural configurations for different forms of mobility: balancing (left), prone-scooting (center), knuckle-walking (right).

Each form of mobility and postural transition is available to the robot as a skill in the form of a close-loop controller. Time and energy-based cost models for these skills based on empirical data are presented in Section 3.3.4. For the experiments in Chapter 6, only balancing and prone-scooting are used for simplicity. Performance specifications are only provided for uBot-6 as the corresponding applications have not been ported to uBot-7 yet.

3.3.1 Balancing Mobility

Balancing mobility combines dynamic balancing in 2-point stance with differential steering (Fig. 3.13). The wheels are controlled by an LQR that maintains the robot's balance while performing translational and rotational movements. The steering geometry allows uBot to rotate in place. This combined with the small 2-point footprint enables navigation in tight spaces. The maximum drive speed and turn rate of uBot-6 in this form of mobility is limited to 1.2 m/s and 4.1 rad/s respectively. Additional torque is reserved for compensatory balancing movements.

3.3.2 Prone-Scooting

In this form of mobility, the robot is configured in a 4-point-elbow postural stance in which the base wheels and the two small wheels attached to the elbows are in contact with the ground (Fig. 3.10(b)). In this mode, the robot drives like a car. However, the steering is more complicated since the front wheels are attached to the elbows and all four arm joints impact the driving behavior. Three shoulder joints with angles θ_1, θ_2, and θ_3 determine both the location and orientation of the elbow wheel for each arm. Figure 3.16 shows the robot while in prone-scooting configuration. Drive inputs include the desired body height, drive velocity, and steering angle.

3.3.2.1 Body Height

In this form of mobility the body height H of the robot is determined by the height of the shoulders h as well as the pose of the forearms. The height of the shoulder is primarily determined by the first shoulder joint angle θ_1. We neglect the minimal influence of θ_2 and θ_3 in order to simplify control. Given a desired body height, the necessary shoulder height is calculated by subtracting a safety distance of 12 cm: $h = H - 0.12\,\text{m}$. The required shoulder joint angle θ_1 is given by

$$\theta_1 = acos(\frac{h - r_1}{b}) + acos(\frac{h - r_2}{a}), \tag{3.1}$$

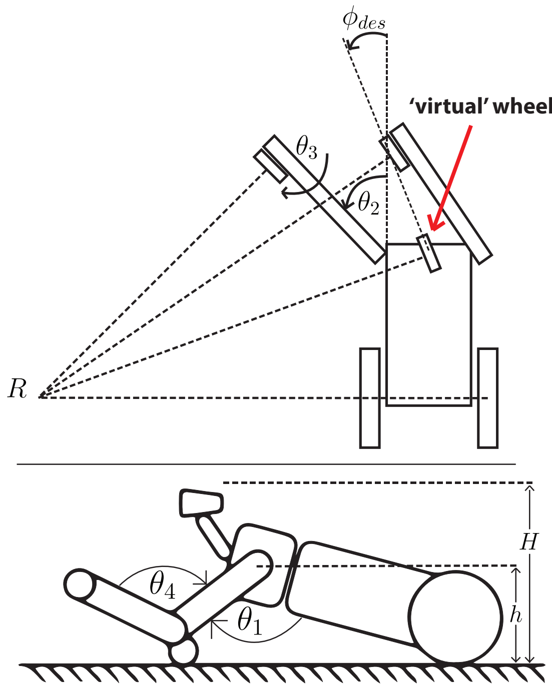


Figure 3.16. Prone-scooting mobility. Top: the desired steering angle ϕ_{des} is used to orient a 'virtual' wheel located midway between the shoulders. The intersection of the axes of the base wheels and the 'virtual' wheel determines the center of rotation R. Control of shoulder joint angles θ_2 and θ_3 allows this turn radius to be matched with both front wheels to avoid slip; bottom: shoulder angle θ_1 and elbow angle θ_4 determine shoulder height h and body height H.

where a is the upper arm length, b is the distance from the base wheel axis to the shoulder, and r_1 and r_2 are the wheel radii of the base and elbow wheels respectively. The elbow joint angle θ_4 only influences the body height if the forearm protrudes higher than the body. In this mode, the body height for uBot-6 can be varied between 24 cm (in which the robot is almost completely prone) and 46 cm.

3.3.2.2 Steering Angle

Scooting requires Ackermann steering where all wheel axes must intersect at a common center of rotation (Fig. 3.17). However, unlike cars, the arm joints influence the location and orientation of the wheel axes of the front wheels. The wheel axes are not guaranteed to be parallel to the ground plane any more. Therefore, instead of the wheel axes we use their projection onto the ground plane throughout the rest of this section.

In order to convert the steering angle input into a turn radius, we define a 'virtual' wheel midway between both shoulders following Cholet *et al.* [25]. The steering angle input specifies its orientation ϕ_{des} (see Fig. 3.16 top). The intersection between wheel axes of the 'virtual' wheel and the base wheels defines the desired center of rotation R.

To ensure that the wheels do not slip, the free shoulder angles θ_2 and θ_3 are controlled such that the wheel axes of the front wheels intersect the desired center of rotation. Depending on both the body height and the current arm configuration, the shoulder joints have varying and coupled influence on the location and orientation of the front wheels.A Jacobian-based controller minimizes the orientation error of the front wheels.

Given the steering angles ϕ_i, where $i = 1$ denotes the left arm and $i = 2$ denotes the right arm, the steering error $\epsilon_i = \phi_{des} - \phi_i$ is computed. The orientation constraint of the front wheels is expressed as a quadratic potential function ϵ_i^2. The error Jacobian J_i that describes the sensitivity of the orientation error with respect to the control variables θ_2 and θ_3 is given by

$$J_i = \begin{bmatrix} \frac{\partial \epsilon_i^2}{\partial \theta_2} & \frac{\partial \epsilon_i^2}{\partial \theta_3} \end{bmatrix}. \tag{3.2}$$

These Jacobians are calculated numerically and allow us to compute updates for reference inputs of the shoulder joints with

$$\begin{bmatrix} \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = -J_i^{\#} \epsilon_i^2, \tag{3.3}$$

where $J_i^{\#}$ is the pseudoinverse of J_i [98]. The resulting closed-loop controller synchronizes both arms such that the desired turn radius is achieved while ensuring that the wheels do not slip.

The minimal turn radius is dependent on the chosen body height. When the robot is at its lowest body height, a turn radius of 1.2 m can be achieved. In this configuration, the steering depends mostly on the second shoulder joint (which is limited to avoid collision with the upper trunk). At its highest body height, a turn radius of 0.68 m can be achieved. The third shoulder joint is at its joint limit preventing tighter turning. Figure 3.17 shows the intersection of the wheel axes and minimal turning radius according to Ackermann steering for a low (left) and high (right) body height. The minimal turn radii achievable for varying body heights is shown in Figure 3.18.

3.3.2.3 Drive Velocity

The desired drive velocity is used to calculate reference velocities for each base wheel. Depending on the desired turn radius of the robot, the base wheels travel on different radii and thus are commanded different velocities in order to prevent slipping. The maximum drive speed for uBot-6 is 1.8 m/s.

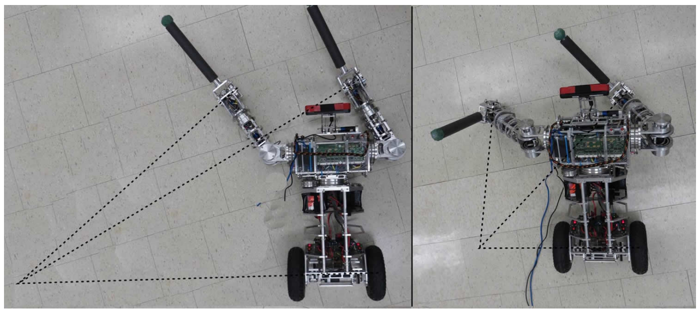


Figure 3.17. Maximal turning for two different robot body heights while prone scooting. The dotted lines depict the wheel axes intersecting at the center of rotation. When the body is almost flat on the ground (left) the turn radius is strongly restricted by the joint limits of the arms. With a raised body, two joints in each arm are used for steering allowing a significantly smaller turn radius (right).

3.3.3 Knuckle-Walking

Knuckle-walking is an upper-limb walking gait that uses the hands as ground contacts. It is based on a sequencing of 3-point and 4-point postural configurations (Figure 3.19). Kuindersma *et al.* presented a preliminary implementation of this controller for uBot-5 [80]. In this work a walking gait was learned efficiently in simulation from primitive controllers.

The knuckle-walking gait provides uBot the ability to traverse otherwise hazardous terrain. This ability comes at the cost of higher energy consumption and slower movement speed compared to locomotion while balancing. The full demonstration of knuckle-walking on the uBot-6 and uBot-7 remains as future work.

3.3.4 Costs

For each skill of uBot-6 the costs with respect to time and energy have been modeled empirically. Knuckle-walking mobility is left for future work as experiments were limited to balancing and prone-scooting mobility.

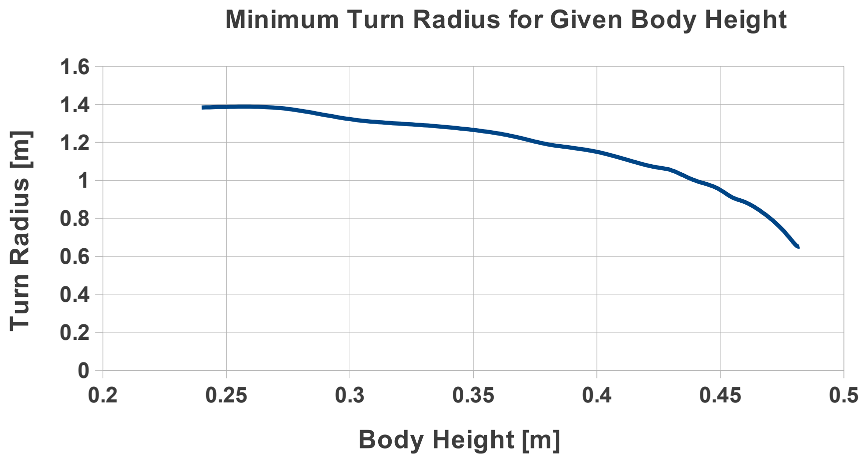


Figure 3.18. Dependency of available minimal turn radius from the body height for uBot-6. With increasing body height, smaller turn radii can be achieved as different joint combinations are used for steering.

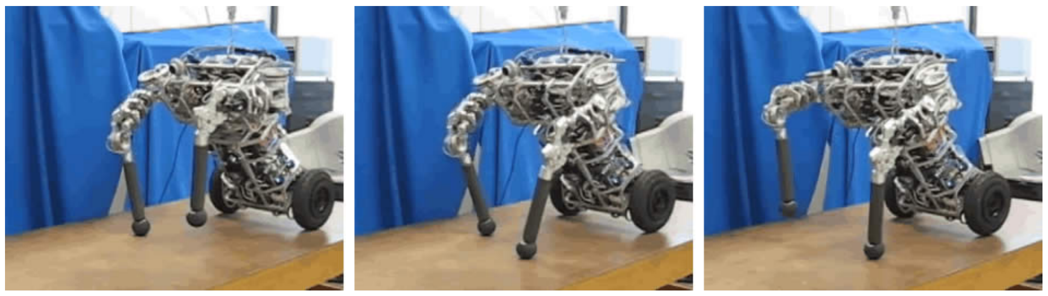


Figure 3.19. Sequencing of 3-point and 4-point postural configurations for knuckle-walking mobility for uBot-5.

3.3.4.1 Balancing mobility

Dynamic balancing is very energy efficient as only small compensation movements have to be performed. While standing in place, the wheels require about 2.6 W to balance. Figure 3.20 shows the variation in power consumption of the wheel motors with drive speed. For the experiments in this chapter, the drive speed for balancing mobility is fixed to a conservative 0.5 m/s for simplicity. The robot turns in place at 0.6 rad/s.

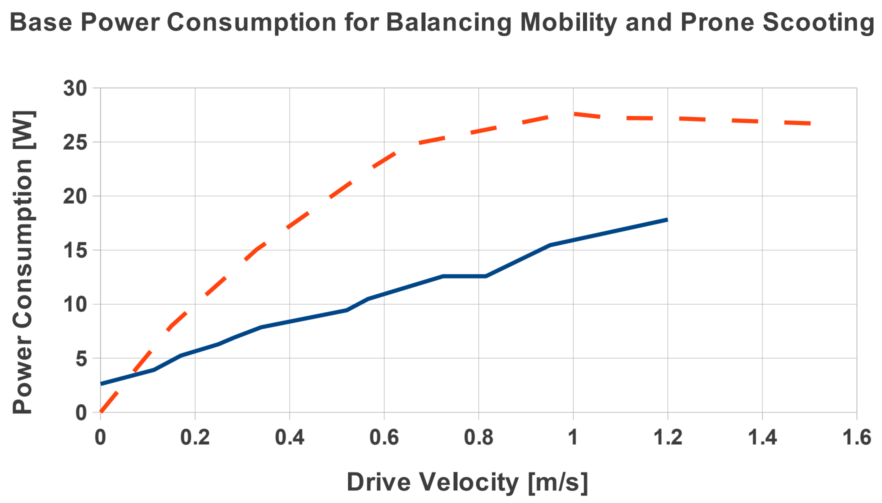


Figure 3.20. Comparison of power consumption [W] of the wheel motors of uBot-6 over drive speed [m/s] in balancing mobility (solid line) and prone-scooting (dashed line). Pure balancing while standing consumes 2.6 W. It should be noted that while prone-scooting the power consumption is dominated by the arms with additional $16 - 80$ W depending on the body height.

3.3.4.2 Prone-scooting

The energy cost for prone-scooting depends on two factors: the power consumption of the arms to keep the body up and the power consumption of the wheels to drive the robot. The power consumption of the wheel motors is similar to that of balancing mobility and ranges from 0 W to 24 W depending on the velocity. The power consumption of the arms to hold up the body ranges from 80 W (almost prone) to 16 W (tallest).

For the experiments in this chapter, the drive speed for prone-scooting mobility is fixed to 0.625 m/s for simplicity. It is slightly higher than balancing mobility as no torque reserves are needed to balance.

3.3.4.3 Postural transitions

Time and energy costs for transitions between postural configurations can be found in Table 3.3. A sequence of several transitions can be necessary in order to switch between forms of mobility.

Table 3.3. Time and energy required to execute transitions between postural configurations for uBot-6.

	Mean energy used (J)	**Mean execution time** (s)
2-point to 4-point (hand)	188.43 ±5.11	5.26 ±0.07
4-point (hand) to 2-point	260.07 ±7.96	6.03 ±0.03
4-point (hand) to Prone	631.57 ±84.52	9.59 ±0.09
Prone to 4-point (hand)	981.42 ±71.05	10.04 ±0.11
4-point (elbow) to Prone	118.10 ±51.35	2.25 ±0.97
Prone to 4-point (elbow)	165.11 ±12.77	4.20 ±0.10

3.4 Manipulation Capabilities

The robots have two 4-DOF arms and a rotatable trunk. The two arms provide a large bimanual workspace that is symmetric in front and back of the robot and includes much of the ground plane (Fig. 3.21). The joint for trunk rotation extends the bimanual workspace all around the robot.

The arms contribute significantly to the center of mass (COM) of the robot. As the robot is a dynamic balancer, a shift in the COM requires an updated setpoint for the balancer to prevent drift of the robot. The COM is continuously calculated from the joint angles and the set point for the balancing algorithm is updated accordingly. The impact of the arm inertia during fast arm movements is handled by a damping term. As a result, the robot will lean its body to compensate for the shift in the COM due to arm movement and will remain stable, but this can result in a shorter reach of the robot's arms.

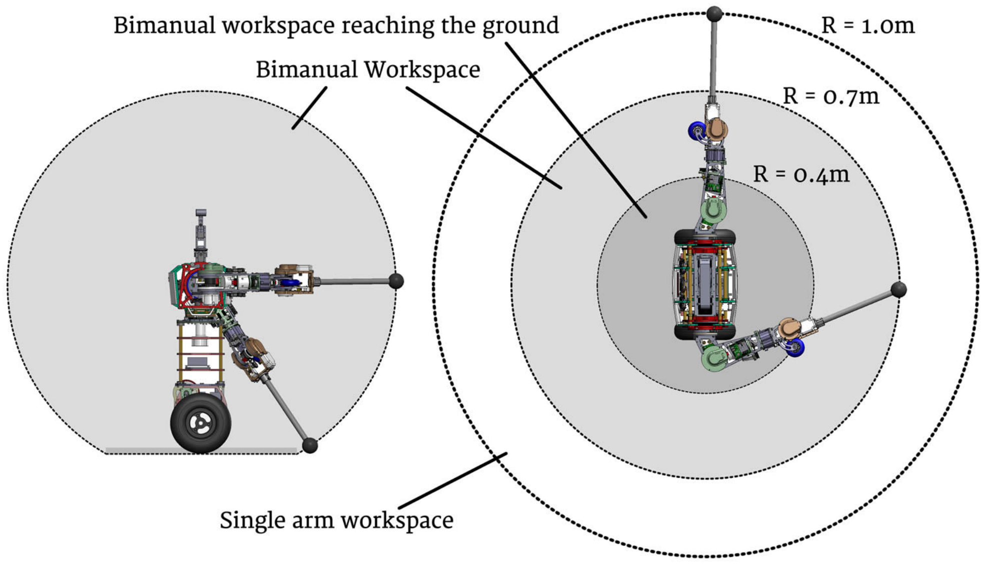


Figure 3.21. Manual workspace of uBot-7: Single arm workspace, bimanual workspace, and bimanual workspace on the ground.

Instead of articulated hands, both uBot-6 and uBot-7 have spherical end effectors with a grippy rubber surface that supports slip-free grasping. The design of suitable hands that support both manipulation and the specific needs for postural transitions is planned future work. Force/torque sensors in the end effectors of uBot-6 enable closed-loop grasp control and detection of failed grasps. uBot-7 adds additional force/torque sensing capabilities through the SEAs in every arm and torso joint.

Wheel motors are typically only assigned to mobility tasks, and arms are used for manipulation. The uBots ignore this boundary. For example, they use the arms to improve mobility by transitioning to alternative postural configurations, and they can use their strong wheel motors to improve performance in tasks such as pushing by exploiting its own body weight [140].

In the past, the uBot series has been used to manipulate indoor environments with switches, buttons, and levers [71]. In this work, several manual skills are used to interact with objects in mobile manipulation experiments (Section 5.2.1).

3.5 Summary

uBot-6 and uBot-7 have been developed as a unique design point combining many advantageous properties into a single robot platform. They are very versatile mobile manipulators that provide the physical capabilities to support several different methods of solving tasks in mobility and manipulation.

uBot-6 introduces several alternative forms of mobility to a humanoid robot. These mobility options are used for the dexterous mobility domain in Chapter 6. Its manual capabilities to interact with the world are used in Chapter 5 to manipulate objects.

uBot-7 is the newest member of the uBot series of mobile manipulators. To our knowledge, uBot-7 is the first wheeled dynamic balancer with series elastic actuators (SEAs). The SEAs will provide increased performance for manipulation tasks, safety mechanisms for the robot in case of falls, and the ability to detect accidental collisions to work around humans more safely. Added degrees of freedom in the head support much better perceptual performance that is independent of postural configurations.

All experiments in this work are based on uBot-6 as the integration of uBot-7 has not been completed yet. Once the integration of uBot-7 is completed and current software has been ported, its overall improved design should increase performance of currently possible applications and add versatility in solving tasks in manipulation and mobility.

CHAPTER 4

BELIEF-SPACE PLANNING

To autonomously perform tasks in the world, a robot usually has to perceive its environment to estimate the state of itself and of the world around it, plan a sequence of actions to reach the task goal, and then try to execute this sequence of actions. This process is constantly repeated during execution to update the state estimate and possibly replan. All of these steps are difficult in real world scenarios due to partial observability, uncertainty of observations and actions, and state transition uncertainty itself. Due to partial observability, a robot is usually not able to observe all relevant features in the environment at once. It can only obtain incomplete 'views' of the world and has to put the information together to get the full picture. As a result, state estimation of both the robot and the world is often imprecise and very ambiguous. The robot does not know for certain from which state to start planning. Additionally, uncertainty in observations can result in false positives or false negatives when detecting features in the environment. This could be due to occlusion, unreliable sensors, or just sensor noise. Even if the existence of environmental features is detected correctly, the location and orientation is still subject to variance. When planning or executing actions, the outcome is usually non-deterministic. This can include multiple possible action outcomes, but also just variance in the precision of the outcome as well as the required time to perform the action.

Belief-space planners handle these issues by using a belief distribution over the robot and world state to guide the planning process. Probability distributions support handling uncertainty in observations and action outcomes. These adjustments usually come at the cost of very slow runtime. As a result, only small problems can be solved in a reasonable time to still be useful on a real robotic system. This makes the tractability of an online belief-space planner very dependent on the structure and representation of the problem domain.

The main contributions in this chapter are:

- ***A compact but expressive state representation for belief-space planning based on object models.*** Existing affordance-based object models, the aspect transition graphs (ATGs), are extended with geometric information. This enables object-centric modeling of features and actions, making the model much more expressive without increasing the complexity. The models provide the state space, transition model, cost estimates, and observation model for belief-space planning.

- ***A method to specify generic single-object tasks in belief-space.*** We introduce a method to express different tasks by defining a task partition over the state space and specifying an information-based metric. We use an object-centric state representation, and thus tasks are specified in the context of single objects. We show how this mechanism supports different task types commonly encountered in robotics.

- ***A planning framework that incorporates the ATG object models and task specifier to handle generic tasks in belief-space.*** We present a belief-space planning framework capable of efficient action selection on a real robotic system under partial observability. The framework also handles uncertainty in action outcomes and in observations. The ATG models provide all information for belief updates and efficiently guide the search of the planner. A hierarchical approach supports efficient planning for scenes with multiple objects.

In the following section, we define the problem formally. Section 4.2 introduces our object and environment models and discusses their impact on belief-space planning. We present our belief-space planning framework that uses the object models in Sections 4.3. Finally, we explore the use of a POMDP solver in combination with the object models in Section 4.4.

4.1 Problem Definition

In general, we want the robot to complete a given task in an environment that is only partially observable and has uncertainty in action outcomes and observations. In most cases many different solutions are available to complete the task successfully. The quality of the solutions can be evaluated with respect to various criteria. Typical metrics in robotics are the number of actions taken, the amount of time required, and the amount of energy expended to reach the goal. In this work we will consider the first two metrics though metrics can be exchanged easily if actions are modeled accordingly.

A popular method to formulate the general problem of selecting best actions in a partially observable environment with uncertain action outcomes and observations is as a partially observable Markov decision process (POMDP). A POMDP is specified by the tuple

$$\langle S, A, T, R, Z, \Phi, \gamma \rangle \, ,$$

where S is the set of state, A is the set of possible actions, $T : S \times A \times S \to [0,1]$ is the set of conditional transition probabilities between states, $R : S \times A \to \mathbb{R}$ is the reward function, Z is the set of possible observations, $\Phi : A \times S \times Z \to [0,1]$ is the observation function, and $\gamma \in [0,1]$ is the discount factor. As the state S in the POMDP is only partially observable, we use the concept of a belief state, b, to represent the probability distribution over states. All possible beliefs or belief points b form the set B. An optimal solution to the problem expressed by such a POMDP is then given by the policy $\pi : B \rightarrow A$ which maximizes the reward for a possibly infinite horizon. The optimal action at a timestep maximizes the expected discounted future reward $E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

In a POMDP, the reward function R encodes the desired metric, such as minimum total execution time, used to evaluate the quality of the solution. In this work we employ belief-space planning methods to solve problems that can be defined as POMDPs, but do not utilize the reward function to encode the evaluation metric.

Chapters 5 and 6 cover experiments in mobile manipulation and dexterous mobility domains respectively, and each uses a different problem description.

In the following sections we will discuss what impact the choices of representation for states, actions, and observations have on solving the problem.

4.2 Knowledge Representation

The capabilities and performance of a planning and reasoning system for robotic systems rely on the representation of the world, the robot, and its own capabilities. A poorly chosen representation can result in high complexity during planning or a lack of expressiveness. High complexity will make even approximate planning intractable for online use on a robot. Lack of expressiveness results in limited capabilities to interact with the environment or to define tasks accurately. For example, the popular grid world domain is simple enough for possibly tractable planning, but it offers almost no usefulness for real robot control [120].

Aspect Transition Graphs (ATGs) have previously been used as object models for belief-space planning. We introduce a powerful extension to the model that provides a more compact representation and at the same time offers higher expressiveness with respect to the capabilities of the robot. The improved model can be used for modeling objects and environments alike.

4.2.1 Aspect Transition Graph (ATG)

Our framework uses a specialized model for objects and environments called an Aspect Transition Graph (ATG) [122]. These models are centered around the long-known concept of aspects [67] and graph-based models of the interaction capabilities with objects [106]. In general, only a subset of the features attributed to an object can be detected from any given sensor geometry [67]. These subsets of features define the *aspects* of the object. In previous work, aspects were used as nodes in a multi-graph where edges represent actions that cause probabilistic transitions between the aspect (Fig. 4.1 and 4.2) [122, 123, 75, 76, 74]. To retain small sizes of action spaces, the actions were either robot-centric or based on visual servoing.

For this work, the ATG model has been improved with several extensions. First, we make a strong distinction between *aspects* and *aspect nodes*: aspect nodes are the nodes in the multi-graph of the ATG, and each aspect node is associated with an aspect defining the set of features that can be perceived together. An aspect, however, can be associated with several different aspect nodes. While these aspect nodes are perceptually identical, their connectivity within the multi-graph is different. For example, if an object has identical front and back sides, there is an aspect node for each front and back of the object where both aspect nodes expect to see the same aspect. Yet turning the object 90° will possibly reveal different observations from the right or the left side. This separation of aspects from aspect nodes has a strong beneficial impact on the complexity of the observation space (see Section 4.2.3.3).

Secondly, we extend the ATG model with geometric information. Features of aspects and actions are specified and parametrized with respect to an object frame [118]. This improves differentiation of aspects since geometric constellations of features are much richer than matching based on bags-of-features. Additionally, the geometric information in the models can be used to predict sensor geometries for new observations and supports better pose estimation. Actions are much more expressive and robust without increasing the complexity. The impact on the state and action spaces is discussed in Sections 4.2.3.1 and 4.2.3.2 respectively. Actions are implemented as controllers with parameters, and these parameters along with estimates of the cost of the action are stored in the ATG. ATG models can be hand-built or autonomously learned by the robot [75, 76, 74, 152]. The models used in this work are hybrids, wherein nodes and edges associated with visual actions were learned, and those for manual actions were hand-built.

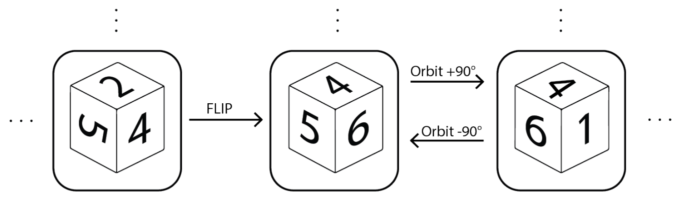


Figure 4.1. An example of actions on transition edges connecting several aspect nodes in an ATG.

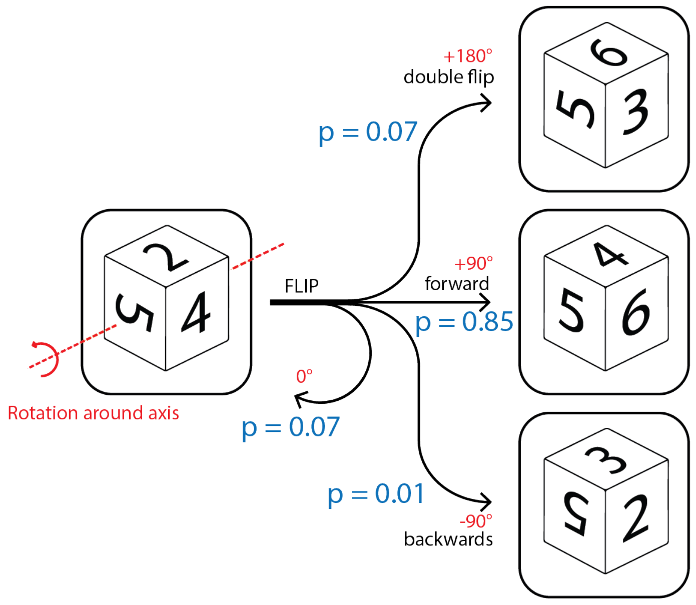


Figure 4.2. An example of uncertain action outcomes in an ATG model: the start node (left) has several outgoing edges for a *FLIP* action. Each edge represents a possible outcome leading to a different aspect node. Probabilities for each transition are stored on the edges.

A Dynamic Bayes Net (DBN) is used as a recursive, hierarchical inference engine in which objects o generate aspect nodes x that then generate observations (aspects) z that can be viewed from a single sensor viewpoint (Fig. 4.3). The DBN fuses the history of observations z and actions a into a maximum likelihood distribution over aspect nodes. The belief $bel(x)$ over the aspect nodes of all known ATG models is used as state in a belief-space MDP. The ATG provides forward models $p(x_{t+1}|x_t, a_t)$ and information for observation models $p(z_t|x_t)$ that are used for the belief update.

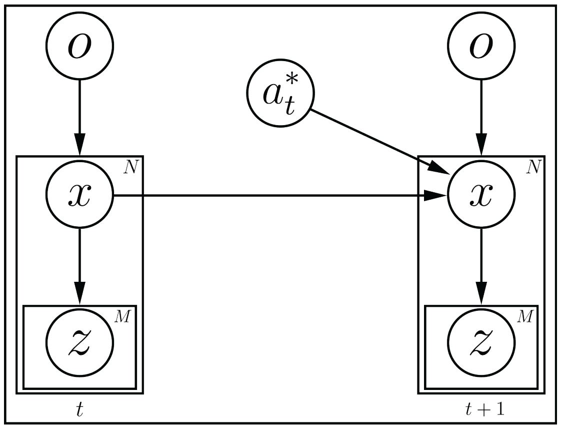


Figure 4.3. Dynamic Bayes Net (DBN) used to model objects and environments. Objects o generate N aspect nodes x that generate M observations z. Actions a are modeled with respect to their influence on aspect nodes x.

As aspect nodes are based on perceivable features of the object as well as the interaction possibilities known to the robot, the aspect nodes of an ATG represent the states of an object that matter to the robot. Using aspect nodes as an abstract state x greatly reduces the state space of the problem (see Section 4.2). The ATG also contains all relevant (known) actions to interact with the object. Therefore, out of all possible action parameterizations, only useful ones provided by the ATG need to be considered. Additionally, the ATG provides forward and observation models for belief update and planning.

As ATGs model objects or the environment with respect to the capabilities of a specific robot, they might be tied to this robot. It is feasible to transfer models completely or partially between robots if they have similar capabilities and will have to be the focus of future work.

4.2.2 Model-Based Aspect Detection

We use perceived features to match model aspects. The features must be of the same *type* (e.g. “visual edge” or “tactile surface normal”), and the geometry must match. There are many ways to match geometric templates. If it is possible to solve the feature correspondence problem easily, then a number of least squares techniques could be used. Alternatively, voting algorithms like RANSAC [41] or Hough transforms [8] can be applied. This work uses the latter.

The aspects detected in the features $\boldsymbol{f}_t$ are computed by matching model aspects (feature types and geometry) to the current perceived features. We would like to estimate the distribution of support for model aspects $p(z_t|\boldsymbol{f}_t)$ as well as the pose at time t of all of the objects that could have generated these observations.

The proposed system extends the generalized Hough transform [8]. Given a random variable representing the distance between features f_i and f_j, $\delta_{ij} = (\mu \in \mathbb{R}, \sigma \in \mathbb{R})$, the area of intersection between the model distance distribution and the observed distance distribution provides a matching score that is used to cast votes for the location of the object frame (Fig. 4.4). The tally over all feature pairs is stored in an accumulator array in $\mathbb{R}^3$ and produces a dominant peak. Using this representation for the aspect geometry, the fully connected graph with all model distance distributions δ_{ij} (Fig. 4.5) are stored in the aspect model.

Algorithm 1 summarizes the Generalized Hough Transform for classifying model aspects in the scene. In Lines (2) and (3), all pairs of features detected in the scene are considered to define possible geometric (sub)structures. In Line (4), samples of the distance between features f_i and f_j are used to approximate a Gaussian inter-feature distance distribution. In the loop defined by Lines (5)–(8), this quantity is compared

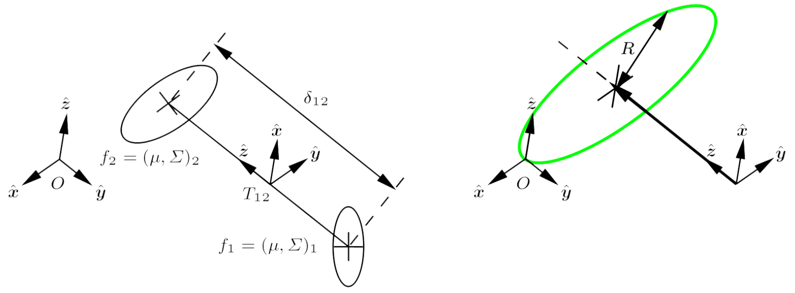


Figure 4.4. Left: a pair of features in the object model with locations and covariances in object frame. Right: resulting Hough voting scheme for a pair of observations matched to these features. Weighted Hough votes are cast on the green circle.

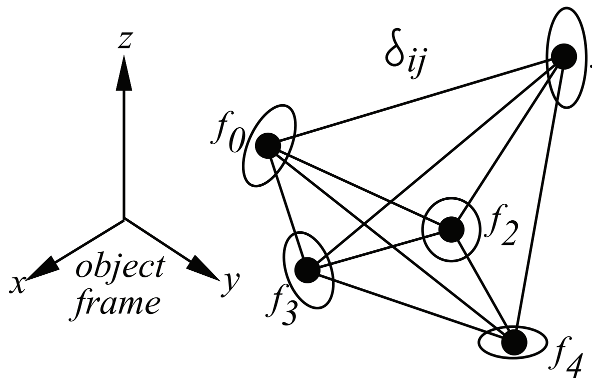


Figure 4.5. The fully connected graph of inter-feature distances used to define an aspect geometry. Each feature is described by its type, mean $\in \mathbb{R}^3$ (dot) and the model covariance $\in \mathbb{R}^{3\times3}$.

to all of the inter-feature distance distributions stored in the aspect models for these features using the Gaussian correlation. The calculated aspects are then used as observations z. The Hough voting mechanism for matching observations to models utilizes the geometric information and supports tolerance to occlusion of features.

Algorithm 1 Aspect-Based Generalized Hough Transform

```
1: initialize aspect models & Cartesian accumulator array
2: for all $0 \leq i <$ NumFeatures do
3:     for all $i + 1 \leq j <$ NumFeatures do
4:         $(\mu_{ij}, \sigma_{ij})$=Sample_Scene_3D_Distance$(i, j)$
5:         for all $0 \leq k <$NumAspects do
6:             if $\langle f_i, f_j \rangle \in x_k$ then
7:                 score $\leftarrow$ score$_{ij}(\mu_{ij},\ \sigma_{ij},\ x_k.\delta_{ij})$
8:                 Cast_Hough_Votes$(i,\ j,\ k,$ score$)$
```

4.2.3 Impact on Belief-Space Planning

The choice of representation can have a large impact on the complexity and expressiveness of a planning method. In this context, we discuss the chosen state, action, and observation representation which is based on ATGs.

4.2.3.1 State Representation

When considering general capabilities of a robot to interact with an object, a large number of actions with continuous multi-dimensional parameters have to be considered. Even typical approaches such as discretization still result in a far too large state space and too many actions to consider for efficient planning.

Let us consider a state representation for manipulation of an unidentified object. The state of the object consists of the type of the object—one of $|O|$ known modeled objects. Additionally, an estimate of the pose q of the object in $SE(3)$ is required to perform actions. The resulting state space is enormous as it is based on the cross product of possible object types with all possible poses. Even for a moderate number of known object models and a reasonably well discretized parameter space for poses, this will result in a very large state space and be prohibitive for planners.

In our framework we use an ATG to model the capabilities of the robot to interact with each known modeled object. The state space is based on an abstracted state of the robot with respect to the object which is represented by aspect node x. The set of all aspect nodes from every known object are denoted as X. The size $|X|$ can be approximated as the number of known object models $|O|$ multiplied by the average number of aspect nodes per object. The pose q of the object is still required to perform actions. The resulting state space is based on the cross product of possible abstract states x and possible poses q: $S = X \times Q$. In general belief-space planning frameworks, the belief could be spread over a large number of states of $X \times Q$. In our framework, based on the structure of the ATGs and the definition of aspect nodes x, a unique pose q can be calculated for each aspect node x from observations. The marginal belief $bel(x_i)$ in just an aspect node x_i is the sum of the belief in that aspect node with all specific poses q_j:

$$bel(x_i) = \sum_j bel(x_i, q_j) \tag{4.1}$$

with $0 < i < |X|$ and $0 < j < |Q|$. As a single pose q_k can be calculated for aspect node x_i, all belief $bel(x_i)$ is concentrated in that combination of aspect node and pose resulting in

$$bel(x_i) = bel(x_i, q_k). \tag{4.2}$$

For all other poses q_j with $j \neq k$ the belief is zero and despite not being handled explicitly, is dealt with correctly and completely. As a result, for each aspect node x there is exactly one pose q that needs to be considered and the effective size of the state space is reduced to $|X|$. All other states of the original state space are still available but have zero probability and do not negatively impact the planner.

4.2.3.2 Action Space

A robot knows all actions in set A with each action a defined by a type and parameters: $A = \mathit{types} \times \mathit{parameters}$. The action parameters can be very high-dimensional to support complex actions available to robots acting in human environments. For example, a simple controller for bimanual grasps could take 3D positions for two hands. Similar to the state space, any discretization still keeping the expressiveness of the actions intact will result in a large number of parametrizations for each action type and thus in a very large size of $|A|$. Some approaches deal with this complexity by using robot-centric descriptions of the actions (e.g. a bimanual grasp is always just happening in a hard-coded position in front of the robot), but this method lacks expressiveness for anything but highly controlled environments. Alternatively, the state description can include that the robot is in the correct pose for such a robot-centric action to work. This would result in a much larger state space.

For belief-space planners, transitions $T(s, a, s')$ need to be known and are typically enumerated. Each time a belief update is performed, all applicable actions need to be considered. This is especially costly when rolling out belief for several steps into the future as the number of actions determines a branching factor in the search tree.

In our framework, for each abstract state the corresponding ATG stores all available actions together with parametrizations in object frame. At runtime, for a state $s_1 = (x, q)$ the available actions can be retrieved from the ATG. To apply process updates, a transition model has to be available for all state action pairs. The ATGs provide an easy mechanism to match an action from a state s_1 to the corresponding action for any other state s_2 based on the pose information of s_1 and s_2. Despite an action space of theoretically infinite size, only few actions have to be stored. This comes at the cost of having to determine corresponding actions at runtime as pose information is not known *a priori*. The structure and representation support calculation of all needed information at runtime for only the few actions which are relevant. This drastically reduces the branching factor in a search tree for planning.

4.2.3.3 Observation Space

In order to perform planning steps to simulate the outcome of actions, all possible resulting observations have to be considered. The number of possible expected observations determines another branching factor in the search tree when rolling out belief over several actions and observations and is one of the limiting factors for belief-space planners. As with state and action space, the observation space is very large with several simultaneously observed features with feature types, 2D or 3D location, possibly orientation, and other continuous variables for other parameters.

We use aspects z_i of an aspect node x_i as observations. Instead of using geometric constellations of features with continuous variables directly as observations, we use a matching mechanism to generate support for perceptual aspects. This limits the number of possible observations to a finite number and enables proper normalization of $p(z_i|s)$ for each state s.

The ATG model provides an efficient way of predicting possible future observations. As a result, only applicable observations can be acquired from the ATG, and only a limited number have to be considered during planning. For each state s the corresponding ATG provides the expected observation z and the probabilities $p(z|s)$ can be precomputed for all states s and stored in the model. As described above, for each aspect node x there is one state s. Therefore, the number of observations $|Z|$ to be considered is equal to the number of aspect nodes: $|Z| = |S| = |X|$.

4.2.3.4 Representation Conclusion

While the state of a robot is usually denoted as s, we use the benefits from our representation in order to simplify our state, action, and observation spaces. Based on this structure, we can refrain from explicitly enumerating some information. We showed above that the probability of $s = (x, q)$ is equal to the marginal probability $p(x)$ over aspect node x. The pose information q is implicitly contained as well but does not have any impact on the equations. Therefore, instead of s we use x as the variable for our abstract state. This changes the observation model to

$$p(z|s, a) = p(z|x, q, a) = p(z|x, a). \tag{4.3}$$

Additionally, for the applications considered in this work, all robot sensors are constantly running. As a result, the observation probability only depends on the state, and we can use

$$p(z|x, a) = p(z|x). \tag{4.4}$$

For the transition model we use

$$T(s, a, s') = T(x, a, x'), \tag{4.5}$$

where a lot of the details are implicitly contained as they are updated at runtime. These choices in representation provide a small effective size of the state, action, and observation spaces without restricting the expressiveness of the framework. The following sections describes how our choices in representation can be used in a belief-space planner.

4.3 Active Belief Planner (ABP)

Our planning framework, called the active belief planner (ABP), uses belief distributions over aspect nodes x to represent the robot's current belief. Each aspect node x is an abstract state of the robot to an object. Bayesian updates are used to incorporate taken actions and new observations into the current belief (Section 4.3.1). The same update mechanism is also used during planning to propagate belief distributions over sequences of multiple actions and observations. We use a hierarchical planning structure to overcome complexity of environments with multiple objects (Section 4.3.2). A separate belief distribution is maintained for each potential object in the environment.

In Section 4.3.3, we introduce a new mechanism to describe tasks in this framework. We provide examples for four task types commonly encountered in robotics (Section 4.3.3.2). Information-based measures can be applied directly to the belief distribution to estimate uncertainty or proximity to a goal (Section 4.3.3.1). We show how such measures are used with the new task representation. Section 4.3.4 presents the algorithm that combines all these methods into our myopic belief-space planner, the ABP.

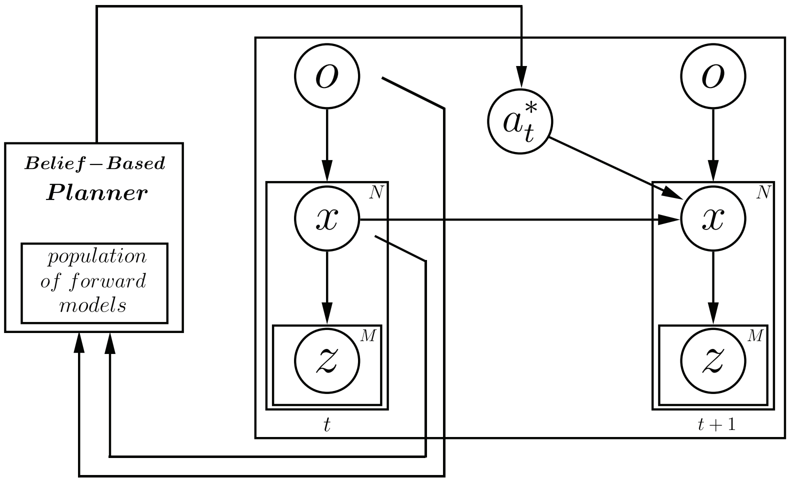


Figure 4.6. The recursive filter for condensing belief in the probabilistic object model. The belief-space planner uses the DBN (Fig. 4.3) to predict how actions *a* cause changes in aspect nodes *x* (and by inference, changes in the distribution over objects *o*) in order to reduce uncertainty in the model space.

4.3.1 Bayesian Update

Our belief-space planning framework tracks and updates belief over the state space. We use aspect nodes x as abstract state (see Section 4.2) of the robot with respect to an object or environment. Belief over aspect nodes $bel(x_t)$ is updated at each time step t based on the executed action a_t and the new observation z_{t+1} using Bayes filtering [142].

The set of known ATGs provides forward models with transition probabilities $p(x_{t+1}|x_t, a_t)$ for all states and actions. In the process update, the belief for every x_t is updated by

$$\overline{bel}(x_{t+1}) = \sum_{x_t \in X} p(x_{t+1}|x_t, a_t) bel(x_t), \tag{4.6}$$

where $\overline{bel}$ denotes that the posterior is due solely to action a_t. In the observation update, the belief for every x_t is updated to incorporate information from new observations z_{t+1} with

$$bel(x_{t+1}) = \eta \; p(z_{t+1}|x) \; \overline{bel}(x_{t+1}), \tag{4.7}$$

where η is a normalizer. The ATG provides all necessary information to establish observations z from perceived features and calculate $p(z|x)$ for any aspect node x.

4.3.2 Hierarchical Planning

In general, tasks can involve multiple objects. Planning over all objects at once can become computationally expensive due to the combinatorial nature of the decision space [24]. Therefore, we cluster features into spatial hypotheses h_k based on the compatibility of their spatial distributions with known object models. The planner can then probabilistically reason over one object hypothesis at a time. The number of hypotheses is expected to be roughly the number of objects in the scene though this is dependent on the quality of the used segmentation mechanism and the complexity of the environment. The complexity of the planning algorithm only increases linearly with the number of independent hypotheses K.

In our framework, we use a Hough transform based mechanism to check compatibility of perceived features with known aspects. This matching uses the geometric constellations of perceived and modeled features. Additionally, the history of perceived features can also be matched against the features of the complete object model.

For spatial hypothesis h_k, the belief distribution is denoted as $bel(x_t^k)$ and observations as z_t^k. Equations 4.6 and 4.7 are extended with hypothesis information to

$$\overline{bel}(x_{t+1}^k) = \sum_{x_t^k} p(x_{t+1}^k | x_t^k, a_t) bel(x_t^k) \tag{4.8}$$

and

$$bel(x_{t+1}^k) = \eta \; p(z_{t+1}^k | x_{t+1}^k) \; \overline{bel}(x_{t+1}^k). \tag{4.9}$$

4.3.3 Task Representation

In earlier work, we used the entropy of posterior belief distributions over objects to select optimal actions for object recognition [118]. We generalize our planning framework to any task that can be expressed as a partition over the set of states (aspect nodes) using a task interpreter [117].

The ABP can plan over any level of the hierarchical DBN (objects, aspect nodes, or features). Assuming a "complete" ATG for all objects in the model space, any task that can be expressed using actions comprising the edges in the ATG can be specified by defining a partition C over aspect nodes of the ATG. This partition aggregates belief on the aspect nodes into targeted subsets for the task. Most tasks result in a partition with two subsets: all aspect nodes that *do* and that *do not* satisfy the task specifications. For other tasks, the aspect nodes may be split into n different subsets to, for example, recognize an object within a model space of n objects.

The belief over the partition C can be calculated by summing the belief over aspect nodes contained in each subset c:

$$bel(c) = \sum_{x \in c} bel(x).$$

We use notation $c(x)$ to denote the specific subset of C an aspect node x belongs to. This mapping from an aspect node x to the corresponding subset c is done in constant time and allows the whole belief aggregation over the subsets of C to be calculated in linear time. This method to express tasks dynamically aggregates belief over subsets of the state space. The underlying belief distributions over the state space are untouched. This supports seamless switching between several tasks without loss of information at little additional cost.

Several tasks can be specified at the same time, each with their own task partition. The planner could evaluate the quality of actions with respect to multiple tasks simultaneously almost without additional cost. Priorities can be assigned to different tasks by weighting their contribution to action utility.

In the next section, we introduce how information-based measures are used to evaluate the belief distributions. Example task types with their respective task partitions are presented in Section 4.3.3.2.

4.3.3.1 Information-Based Measures

Standard information-based metrics can be applied in a belief-space planner to choose the next best action. The planner can perform Bayesian updates to estimate the effect actions and expected future observations can have on the belief distribution. The information-based metric can be used to determine the gained information from a belief distribution to its expected successor. The choice of the metric changes the behavior of the robot. For example, minimizing the entropy,

$$H(c_t) = -\sum_{c_t} bel(c_t) \log\left(bel(c_t)\right), \tag{4.10}$$

causes the belief-space planner to pick actions that efficiently condense belief into the subset c that best represents the history of observations. If the model space contains the correct object, this corresponds to a recognition task. Alternatively, a target distribution $T(c)$ can be specified over all c. In this case, minimizing the Kullback-Leibler (KL) divergence [82] between $T(c)$ and the current belief $bel(c_t)$,

$$D_{KL}(T(c)||bel(c_t)) = \sum_{c_t} T(c) \log \left(\frac{T(c)}{bel(c_t)} \right), \tag{4.11}$$

results in actions that steer the robot toward the target state(s) while automatically balancing information gathering actions and actions towards the task goal. Tasks defined this way are most general and can include recognition at the object and aspect node levels.

In the following sections, the utility of actions is evaluated based on the expected change in the chosen information-based metric. For the remainder of this work, this expected change is generally referred to as 'information gain' IG irrespectible of which information-based metric is used specifically. We use $M(c_t, T(c))$ as place holder for the information-based metric (e.g. entropy or KL divergence) employed to evaluate the belief distribution over the task partition C.

4.3.3.2 Examples of Task Types

The proposed approach can represent a large number of tasks and task types. In this section, we define four basic task types commonly found in robotics problems. These are only samples of possible tasks that can be represented in this framework; tasks are only limited by the expressiveness of the known ATG model. Each type can be differentiated by the way the task partition defines the task for the planner. A graphical example of task partitions for the four task types is shown in Figure 4.7. Demonstrations of these tasks are shown in Sections 5.4.1 and 5.4.2.

4.3.3.2.1 Recognition Task In a *recognition* task, the robot is presented with one or more object(s) of unknown identity. The robot has ATG models for n different objects and has to identify the probability that the data supports each of the known

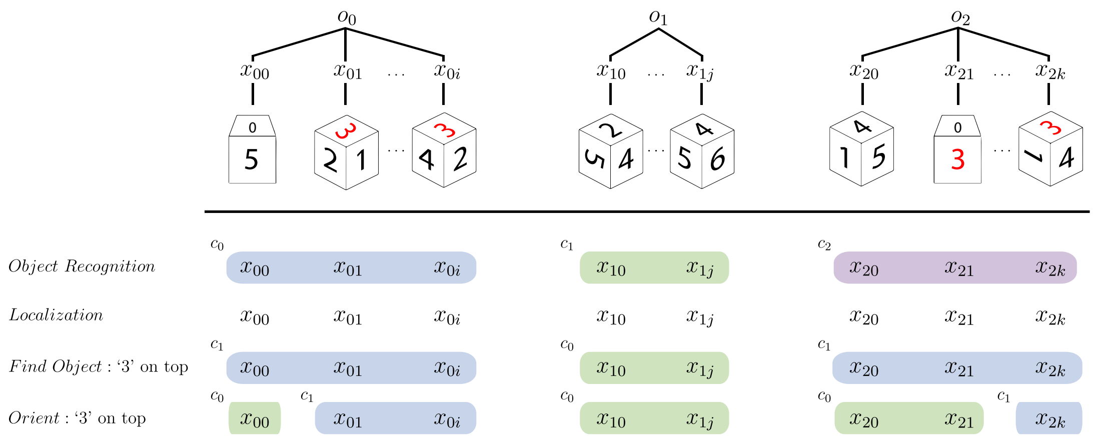


Figure 4.7. A simplified DBN for three objects is shown here. Task partitions for examples of each task type are shown below. For the *recognition* task, all aspect nodes for each object are grouped into one subset of the partition, resulting in three subsets in the partition (colored blue, green, and purple). For the *localization* task, the task partition contains single-element subsets, where the element is one aspect node from one of the three objects, resulting in $(i + j + k + 3)$ subsets (uncolored for readability). For the *find* task, we show an example of finding an object with a '3' feature on top. The task partition for this contains two subsets: one has all the aspect nodes of two objects where at least one aspect node has the '3' feature on top (colored blue) and the second has all the remaining aspect nodes (colored green). Finally, for the *orient* task, we show an example for orienting an object such that the '3' feature is on top. This task partition also contains two subsets. One has all the aspect nodes where the '3' feature is on top (colored blue) and all remaining aspect nodes belong to the second (colored green).

ATG models. The robot can use any action present in all of the ATGs to investigate and manipulate the object(s). The goal is to condense belief into a single subset of the task partition defined by objects in the model space. In other words, if all objects in the scene should be identified, the belief for each hypothesis h_k must condense on one object identity. This task type can be expressed mathematically as

$$\forall h_k \; [\max_j bel(c_j^k(x)) > \beta], \tag{4.12}$$

where β is some threshold for the belief. The task partition C over all aspect nodes from all ATGs splits the aspect nodes of each ATG model into a separate subset, resulting in a partition with n different subsets c_j:

$$c_j = \{x_i | p(o_j | x_i) = 1\} \quad \text{for} \quad 0 \leq j < n.$$

The row in Figure 4.7 labeled 'Object Recognition' illustrates this partition.

4.3.3.2.2 Localization Task A *localization* task establishes the pose with respect to features of one or more object(s) encoded in aspect nodes. The robot is presented with a single sensor view of an object with either known or unknown identity. For each hypothesis, the robot has access to $|X|$ aspect nodes for all n ATG models and has to identify which known aspect node x_i, $0 \leq i < |X|$, corresponds to the constellation of features detected in this single view. Again, the robot can use any action available in the ATG models to investigate and manipulate the object(s). This task type has the same mathematical formulation as *recognition* (Eq. 4.12) with a different task partition. The task partition C for *localization* divides each aspect node for all ATG models into separate subsets, resulting in a partition with $|X|$ different subsets c_j:

$$c_j = \{x_j\} \quad \text{for} \quad 0 \leq j < |X|.$$

Once belief is condensed on an aspect node, the robot knows which object it is sensing and where it is relative to that object. An example of a resulting partition can be found in the row labeled "Localization" of Figure 4.7.

4.3.3.2.3 Find Task Often the specific identity of object(s) or aspect node(s) is not important. Instead, the utility of an object for a task can be based on a subset of its properties such as visual appearance, haptic responses, or interaction possibilities. Thus, a robot can be asked to *find* a suitable object—one that contains at least one aspect node that satisfies the task specifications.

We define the *find* task as follows: the robot is presented with one or more object(s) of either known or unknown identity and has access to n known ATG models. The robot interacts with the object(s) until it is certain that at least one object satisfies the task specifications. This can be expressed mathematically as

$$\exists h_k \; [bel(c_1^k) > \beta]. \tag{4.13}$$

The task partition C splits all aspect nodes into two subsets—suitable (c_1) and not suitable (c_0):

$$\begin{aligned} c_1 = \{x_i | \exists x_j \exists o_k \; [\; & p(o_k|x_i) = 1 \wedge \\ & p(o_k|x_j) = 1 \wedge \\ & y(x_j) = 1 \;]\}, \end{aligned} \tag{4.14}$$

$$c_0 = X \setminus c_1 \tag{4.15}$$

with

$$y(x) = \begin{cases} 1 & \text{aspect node } x \text{ satisfies task} \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

Both reduction of entropy or KL divergence over $bel(c(x))$ work as metrics to guide the planner. Given one unknown object, the planner determines the suitability of the object for this task. If presented with more unknown objects, it will investigate the most promising object(s) first in order to find a suitable one. An example of a resulting partition can be found in the row labeled "Find" of Figure 4.7.

4.3.3.2.4 Orient Task The *orient* task is a *find* task with the added specification of the configuration that the object should have with respect to the robot. It uses the same mathematical formulation as in Equation 4.13. The same function $y(x)$ from the previous task (Eq. 4.16) is also used, but only matching aspect nodes x are considered as task success (as opposed to all aspect nodes of objects with at least one matching aspect node as in Equation 4.14):

$$c_1 = \{x_i | y(x_i) = 1\}, \tag{4.17}$$

$$c_0 = X \setminus c_1. \tag{4.18}$$

By selecting actions that condense belief in c_1 using KL divergence as the metric, the robot can manipulate objects into a desired configuration to satisfy task requirements without having to know the precise identity of the object. An example of a resulting partition can be found in the row labeled "Orient" of Figure 4.7.

4.3.4 Myopic Belief-Space Planning

An overview of the model-based ABP is shown in Figure 4.6. The planner maintains a belief distribution over aspect nodes $bel(x)$ for each spatial hypothesis h_k. It uses a forward model to expand a search tree through several candidate actions and score future belief states using information theoretic measures (Section 4.3.3.1).

In principle, the search tree can be expanded to any depth, however, without an auxiliary policy structure, the size of the tree and the planning time are prohibitive.

We propose that expanding the search tree to a low depth, often even depth 1, provides useful guidance for many tasks. The time required to expand all belief nodes is dependent on the distribution of belief and quickly decreases when the belief condenses on fewer aspect nodes. The search depth of the algorithm is variable and is automatically increased as belief condenses and forward planning becomes less expensive. For simplicity, the resulting algorithm is shown for a 1-ply search in Algorithm 2.

Algorithm 2 Active Belief Planner (shown for 1-ply)

```
1:  $\alpha$ = Future observation update threshold
2:  $\tau_{h_k,a_t} = 0$ for all $h_k, a_t$
3:  $\boldsymbol{IG} = \{\}$
4:  for all $h_k$ do
5:    for all $a_t$ available in ATG do
6:      $bel(c^k_{t+1}) = 0$ for all $c^k_t$
7:      for all $x^k_{t+1}$ do
8:        $\overline{bel}(x^k_{t+1}) = \sum_{x^k_t} p(x^k_{t+1}|x^k_t, a_t) bel(x^k_t)$
9:      for all $x^k_{t+1}$ do
10:       if $\overline{bel}(x^k_{t+1}) > \alpha \max_{x^k_{t+1}} (\overline{bel}(x^k_{t+1}))$ then
11:         $z^k_{t+1} \leftarrow ATG(x^k_{t+1})$
12:         for all $x^k_{t+1}$ do
13:           $bel(x^k_{t+1}) = \eta\, p(z^k_{t+1}|x^k_{t+1})\, \overline{bel}(x^k_{t+1})$
14:           $bel(c^k_{t+1}(x))$ += $bel(x^k_{t+1})$
15:         $m = M(c^k_{t+1}, T(c))$
16:       else
17:         $m = M(c^k_t, T(c))$
18:       $\tau_{h_k,a_t} = \tau_{h_k,a_t} + \overline{bel}(x^k_{t+1}) m$
19:     $IG_{h_k}(c^k_t, a_t) = M(c^k_t, T(c)) - \tau_{h_k,a_t}$
20:     $\boldsymbol{IG} = \boldsymbol{IG} \cup IG_{h_k}(c^k_t, a_t)$
21: while arg $\max_{h_k,a_t} \boldsymbol{IG}$ is not feasible do
22:   $h^*_k, a^*_t$ = arg $\max_{h_k,a_t} \boldsymbol{IG}$
23:   $\boldsymbol{IG} = \boldsymbol{IG} \setminus IG_{h^*_k}(c^k_t, a^*_t)$
24: return arg $\max_{h_k,a_t} \boldsymbol{IG}$
```

For each object hypothesis h_k and available action a_t, the algorithm performs a control update to calculate the expected belief $\overline{bel}(x^k_{t+1})$ after taking action a_t (Line 8). Transition probabilities for the process update $p(x_{t+1}|x_t, a_t)$ are stored in the edges of the ATG. We use threshold $\alpha \max_{x^k_{t+1}} (\overline{bel}(x^k_{t+1}))$ (relative to the highest current belief) to exclude beliefs that come from expected aspect nodes with low probability (Line 10). The α term is a single value from range $(0, 1]$ set by the user at the beginning (Line 1). For all experiments, we used $\alpha = 0.1$. The aspect geometry inside the ATG provides an expected observation z_{t+1} for each expected future aspect node x_{t+1} (Line 11). After performing an observation update following Equation 4.9 (Line 13), the belief over the corresponding subset of the task partition $c^k_{t+1}(x)$ is updated (Line 14). The expected information gain IG is calculated for each object hypothesis and action combination (Lines 15–19) with $M(c^k_t, T(c))$ denoting the place holder for the information-based metric employed (e.g. entropy or KL divergence). The action with the highest expected information gain is chosen. The motion planner of the robot examines the feasibility of the selected action, and if the action is not feasible, the planner selects the next best action.

The complexity of the planning algorithm for a greedy 1-ply plan is $O(|A||X|^2)$ where A is the set of eligible actions and X is the set of aspect nodes. The most expensive step in the planner is the measurement update of all aspect nodes x_{t+1} with each expected observation z_{t+1} inside the prediction calculation. But since z_{t+1} is only dependent on the models, all values for $p(z_{t+1}|x_{t+1})$ can be precomputed to greatly reduce computation cost. The complexity is linear in the number of spatial hypotheses h_k.

Additionally, the influence of expected observations z^k_{t+1} on the expected information gain for an action a_t is minimal if they are generated from expected aspect nodes x^k_{t+1} with low probability. Excluding such x^k_{t+1} from the prediction calculation can reduce computational load while having negligible impact on the prediction result. We use threshold $\alpha \max_{x^k_{t+1}} (\overline{bel}(x^k_{t+1}))$ (relative to the highest current belief) to exclude such belief from the prediction calculation. As a result, the complexity of the problem is reduced to $O(|A||X||X'_{t+1}|)$ where $|X'_{t+1}|$ is the number of aspect nodes that have belief above threshold.

Typically, belief is initially distributed over a large number of aspect nodes, and even greedy search can take a long time after the very first observation. We observed that, even for a model set of highly similar objects, the number of aspect nodes decreases very quickly with every action.

4.3.5 Costs: Number of Actions, Time, Energy

A task is specified through a task partition (Sec. 4.3.3) and an information-based metric (Sec. 4.3.3.1). The metric guides the action selection towards more useful or informative actions. If the information gain is used directly for action selection, then the robot tries to minimize the number of required actions to complete the task. Alternatively, the information gain can be weighted by the required time or energy of actions in order to minimize required time or expended energy for task completion. For the experiments and demonstrations in this work, we minimize the number of required actions or the time required to complete the task. An estimate for the cost of actions is stored in the ATG models for all actions (Section 4.2.1).

4.4 Point-based Value Iteration (PBVI)

There are many established solvers for POMDPs, but limited scalability is a known problem. Here, we are using point-based value iteration (PBVI) as an representative for point-based online solvers for POMDPs [103]. It generates a finite set of belief points $B = \{b_0, b_1, \ldots, b_q\}$ and calculates an approximate solution for these belief points only. The belief points are preferably sampled such that planning is concentrated on the most probable beliefs. The algorithm can alternate between belief point expansion and value iteration to achieve anytime properties. Value iteration finishes quickly when the number of belief points is low at the beginning. With increasing size of B, the value iteration will get slower, but the calculated policy will improve.

The algorithm initializes a separate α-vector for each selected belief point and repeatedly updates them via value backup. After n iterations the α-vectors $V_n = \{\alpha_0, \alpha_1, \dots, \alpha_q\}$ provide the current best solution. Each α-vector represent a $|S|$-dimensional hyper-plane and provides the value function over a bounded region of belief: $V_n(b) = max_{\alpha \in V_n} \sum_{s \in S} \alpha(s)b(s)$. Each α-vector is associated with an action. For a given belief b, the value function can be evaluated, and the α-vector resulting in the highest value indicates the action with the highest expected reward. This action is the optimal action for the current belief assuming optimal behavior for all following actions.

As PBVI maintains a full α-vectors for each belief point, the piece-wise linearity and convexity of the value function is preserved and defines a value function over the entire belief simplex (Fig. 4.8).

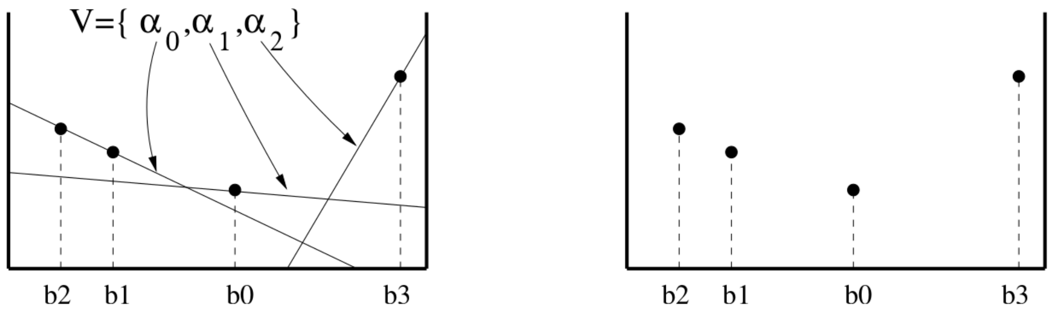


Figure 4.8. Value function representation with full α-vectors for each belief point (left) and just the value for each belief point (right) [103].

4.4.1 Value Iteration

The point-based value backup $V_n = \widetilde{H} V'_{n-1}$ is done with backup operator $\widetilde{H}$. We calculate intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,z}$ for all actions ($\forall a \in A$) and all observations ($\forall z \in Z$):

$$\Gamma^{a,*} \leftarrow \alpha^{a,*} = R(s,a), \tag{4.19}$$

$$\Gamma^{a,z} \leftarrow \alpha_i^{a,z} = \gamma \sum_{s' \in S} T(s,a,s')\Phi(z,s',a)\alpha_i'(s'), \quad \forall \alpha_i \in V'. \tag{4.20}$$

Transition function $T(s,a,s')$ and observation function $\Phi(z,s',a)$ are provided by the ATG object model. Next, we can construct for all belief points ($\forall b \in B$) and all actions ($\forall a \in A$):

$$\Gamma_b^a = \Gamma^{a,*} + \sum_{z \in Z} \underset{\alpha \in \Gamma^{a,z}}{\arg\max}(\alpha \cdot b). \tag{4.21}$$

Then the α-vectors for each belief point is updated with the Γ_b^a for the best action:

$$V \leftarrow \underset{\Gamma_b^a, \forall a \in A}{\arg\max}(\Gamma_b^a \cdot b), \quad \forall b \in B. \tag{4.22}$$

The complexity of the point-based backup operations is only $O(|S||A||V'||Z||B|)$ in contrast to the full backup operation with $O(|S|^2|A||V'|^{|Z|})$. The size of V remains constant and is at most $|B|$. Based on our belief representation and object models, the size of Z is at most $|S|$ with a unique observation per abstract state. The complexity is $O(|S|^2|A||B|^2)$. In practice, states and respective observations with zero belief can be skipped and incur no cost and $|S|$ is rather small.

4.4.2 Belief Point Set Expansion

PBVI performs best when the belief point set is uniformly dense in the set of reachable beliefs. Therefore, we generate the set of belief points by repeatedly simulating a single step forward trajectory to reach other close by beliefs starting from the current belief.

After initializing B with just the current belief, for each $b \in B$, we stochastically simulate single steps for each action $a \in A$ to produce new beliefs $\{b_{a_0}, b_{a_1}, \dots\}$. For each new belief b_{a_i}, we throw it away if it already exists in B. Then we add the b_{a_i}

to B that is the furthest away (L_1 distance) from all other beliefs in B. Typically, at each expansion step, the algorithm chooses the single furthest successor for each current belief point $b \in B$, thus at most doubling the size of B.

4.4.3 Usage with ATGs

We use PBVI in conjunction with ATGs. State representation and Bayesian update are used as detailed in Section 4.3.1. The ATGs for known objects provide all the required information for the transition model and the observation model, but unlike typical POMDP descriptions, actions are not uniquely indexed. As states are abstracted with implicit pose information only known at runtime (Section 4.2.3.1) and actions that are parametrized with respect to object frames, action correspondence between states has to be resolved at runtime (see Section 4.2.3.2). The ATG provides transition models for all actions for all states, but matching an action a for state s_1 to the corresponding action for state s_2 requires pose information which is only available at runtime.

Task specifications such as the task partition introduced in Section 4.3.3 cannot be used directly, but can inform the values of the reward function $R : S \times A \rightarrow \mathbb{R}$. In addition to the states and actions from the ATGs, we introduce an 'at_goal' action to declare that the task is completed and a *heaven* and a *hell* state. If the 'at_goal' action is executed in a state that is part of the goal subset of the task partition, then this results in a transition to the *heaven* state, otherwise to *hell*. All actions executed in these two states self-loop and incur zero cost in case of *heaven* and the maximum cost in case of *hell*. Additionally, the α-vectors can be initialized appropriately for all known goal states from the task partition. This reduces the required search horizon to find a good solution.For tasks that only aim at reducing uncertainty, such as *recognition*, *localization*, and *find* tasks, the reward function depends on the actual belief distribution $bel(s)$ instead of just a goal state-action pair. As value iteration propagates the reward of states, this method is not suitable for these cases by itself. Araya *et al.* have extended POMDPs with belief-dependent rewards [3], but this is left for future work.

Previously, scenes with multiple object have been handled by POMDPs in a hierarchical manner [24, 133]. The use of hierarchical handling of multiple objects (Section 4.3.2) has not been tried with PBVI in this work as PBVI is most useful for *orient* tasks when a specific state needs to be reached. Usually this happens once a specific object in the scene has already been chosen. The information-based metrics for other task types often depend on the actual belief distribution, but the actual distribution is not considered/available in the value iteration step of PBVI as it backs up the reward from successor states independently of the full belief distribution. Additionally, the complexity of PBVI is very prohibitive unless the belief has already been condensed to a small number of states.

4.5 Summary

We presented a new version of the aspect transition graph (ATG) extended with geometric information. ATGs are as object and environment models and form the basis of the knowledge representation for belief-space planning in our framework. We use aspect nodes as abstract state and discussed the resulting beneficial properties on state, action, and observation space. Complexity is reduced while a good expressiveness is maintained. We presented our belief-space planning framework: the active belief planner (ATG). It uses the representation based on ATGs together with a new method to represent tasks for belief-space planners. Objects in scenes are handled hierarchically resulting in near linear complexity in the number of objects. We also presented how an online POMDP solver, point-based value iteration (PBVI) can be used with the ATG-based representation.

CHAPTER 5
MOBILE MANIPULATION EXPERIMENTS

In order to demonstrate the capabilities of the belief-space planning framework, we use the uBot-6 robot to perform various experiments. The experiments are all centered around the new **ARcube domain** for mobile manipulation. The domain has been designed to be extremely challenging for planners and requires integration of multi-modal information.

5.1 ARcube Domain

We introduce the new *ARcube domain* that is based on a model set specifically designed to stress planners. Objects in this set are *ARcubes*, which are rigid cubes with a single ARtag centered on each of the six faces (Fig. 5.1). These boxes can be easily replicated and their size can be adjusted to meet the requirements of different robot platforms. The ARtags can be easily identified and localized and are used as a proxy for more general purpose visual processing [62]. Though ARtags provide id and the full pose of tags, we only use their id and position. By discarding orientation information of tags, several features need to be perceived to establish the pose of an ARcube object. Additionally, the visual ambiguity between ARcubes is much increased as the orientation of tags cannot be used to differentiate them.

The permutation of unique tags applied to objects provides a simple means of controlling the complexity of recognition experiments. The number of visually differentiable ARcubes in the model set given n unique ARtags is $|\mathcal{M}| = \binom{n}{6} \cdot 5 \cdot 3!$, which makes it easy to create very large model set sizes. This large number of possible visually similar cubes and the natural sparseness of visual features leads to a large degree of ambiguity. Additionally, each visually unique cube can have up to six eccentrically weighted counterparts, which are visually identical and can only be differentiated through the transition dynamics of manual actions (Fig. 5.2).

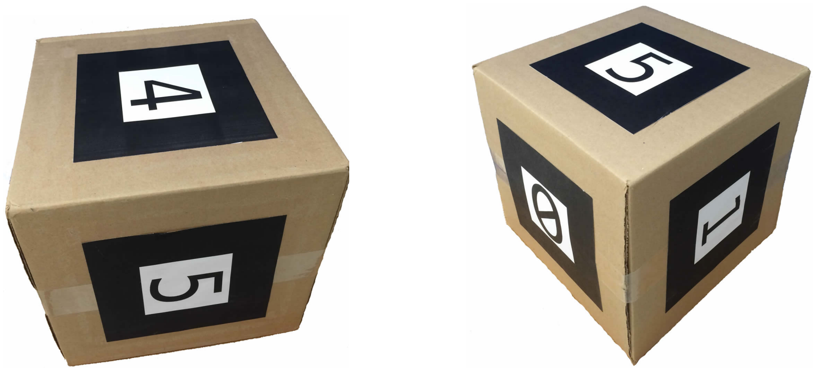


Figure 5.1. Aspects with two features (left) and three features (right) for an ARcube object.

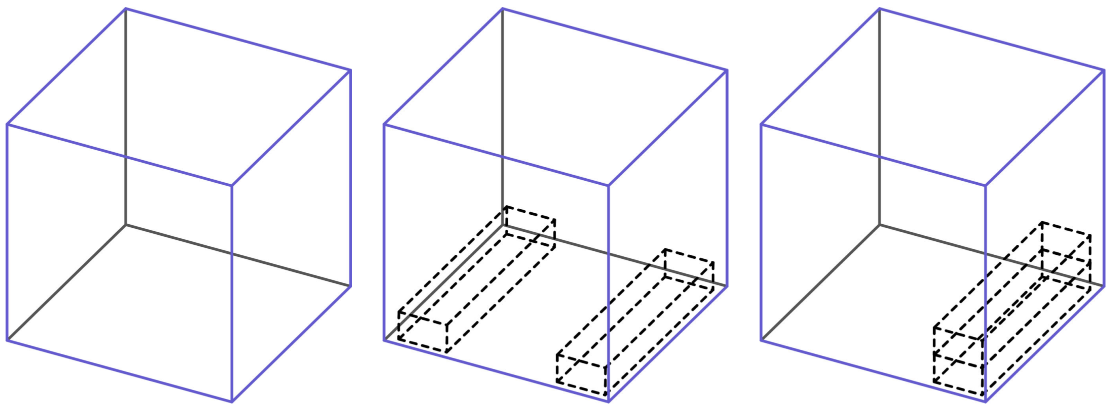


Figure 5.2. Differently weighted variants of ARcube objects: evenly distributed weight (left), eccentrically distributed weight at a face (middle) and an edge (right).

More realistic objects are often easier to differentiate from even a single view. By using ARcube objects, we can generate a domain that is difficult for the planner instead of the perception. The partially observable nature of the ARcubes together with the high ambiguity require long sequences of actions to recognize objects or manipulate them. Experimental recognition tasks constructed this way are very challenging, requiring multiple sensor geometries to fully disambiguate objects within a set of test objects. Furthermore, the efficiency of a sensing strategy varies based on the composition of the model set.

5.2 Experimental Setup

We are using the uBot-6 mobile manipulator (Section 3.1.1) to perform various tasks involving one or multiple ARcube objects. The toddler-sized robot can manipulate ARcubes with both arms and use the mobile base to reposition itself around it. The robot only uses balancing mobility for these experiments.

The robot has four different actions modeled to interact with ARcubes: Flip, Lift, Push, and Orbit. These actions are described in the next section. Details on the ATG models for the ARcubes are shown in Section 5.2.2.

5.2.1 Information Gathering Actions

The following parametrized skills are implemented as information gathering actions to interact with ARcube objects:

1. Orbit – A locomotive action in which the robot drives and reorients itself toward the target object, thus, changing the viewpoint. The robot circles around the object location by an angle $-180° \leq \theta < 180°$ (with respect to the world $\hat{\boldsymbol{z}}$-axis).

2. Push – The robot uses its arm to push along the normal of a chosen face of an ARcube. If necessary, the robot will first reposition itself by driving to a location in front of the face. Pushing causes different outcomes depending on the mass distribution—the ARcube will either move approximately straight ahead causing no visual change or pivot approximately 45° about the world $\hat{\boldsymbol{z}}$-axis, revealing a three-feature visual aspect. The robot receives a suitable drive goal as well as a start location and push direction as parameters.

3. Lift – The robot uses both arms to grasp the object above the geometric center. If necessary, the robot will first reposition itself by driving to a location in front of the face. This will ensure the grasp goals are within reach. The robot will then raise the object and place it back down. For a uniformly weighted object, this sequence of actions causes no reorientation or visual change. Given an object with eccentric mass, the action outcome depends on the location of the center of mass. For example, if the center of mass is at the top-most face of the cube, the action results in the object flipping 180° around the axis between the hands such that the eccentric mass is now at the bottom. Action parameters are a suitable drive goal and two grasp locations for the left and right hands.

4. Flip – Like Lift, the robot grasps the box with both hands, raises it, and puts it back down. Again, this action can be preceeded by a drive action to position the robot better. In contrast to Lift, the grasp goals are closer to the body of the robot near the front face of the box. For an ARcube with uniform mass distribution, this most likely results in a box rotation around the grasp locations, moving the front face to the top. This reveals the previously bottom face of the object in the front. An example illustrating the different outcomes of a Flip action can be seen in Figure 4.2. For eccentrically loaded objects, the action outcomes depend on the location of the center of mass. If the center of mass is not on a side face, the most likely outcome will be a rotation such that the center of mass moves to the base of the object. Action parameters are a suitable drive goal and two grasp locations for the left and right hands. An example of uBot-6 executing a Flip action can be seen in Figure 5.3.

5.2.2 ARcube ATG Models

The robot has access to an ATG model for each known ARcube, though the set of ATG models considered varies in each experiment. There are 20 different visual

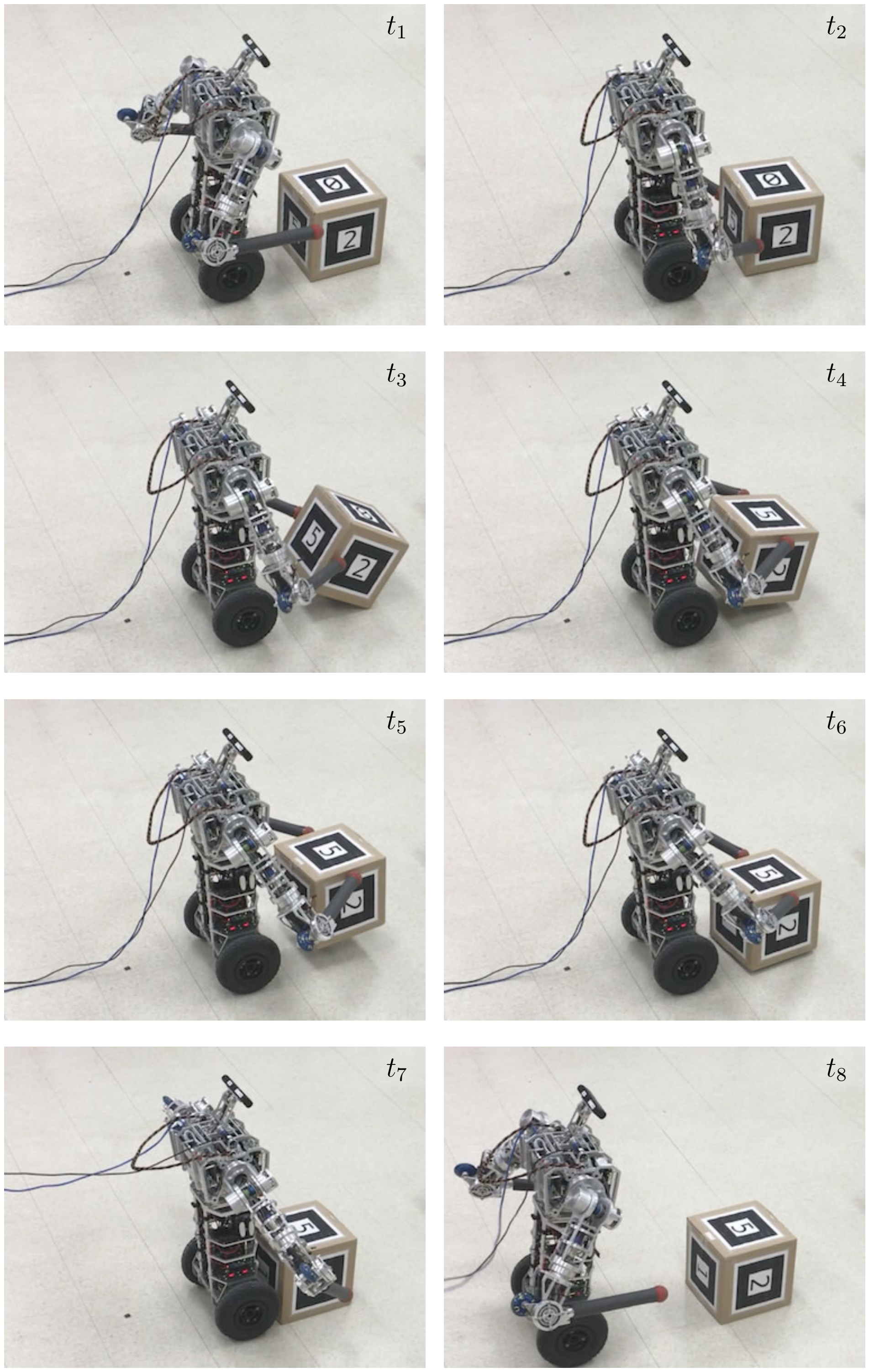


Figure 5.3. The uBot-6 performing an information gathering Flip action on a target ARcube with uniform mass distribution. The action causes a 90° rotation of the ARcube.

aspects that can be found on an ARcube with six non-repeating ARtag features: twelve aspects with two tags, eight aspects with three tags. Aspects with single features are ignored as uninformative. Based on the actions modeled for ARcubes, the orientation of the aspect with respect to the robot is important for action outcomes and thus result in additional aspect nodes. For example, an aspect with two visible tag features can be seen in Figure 5.1 (left). The geometric constellation of the 4 and 5 tags is the same whether 4 is on top and 5 in front or vice versa, but flipping the box results in different transitions for both cases. As a result, each two feature aspect is associated with two different aspect nodes, and every three feature aspect (Figure 5.1 (right)) is associated with three different aspect nodes. In total, each ATG model currently has 48 aspect nodes. When viewing an ARcube square-on, we can refer to the locations of the tags for the current aspect node with 'top', 'front', 'right', 'back', 'left', and 'bottom'. These locations are used for task definitions in the experiments below.

The ATGs for the ARcube objects model action outcomes for four action types: Flip, Lift, Push, and Orbit. These actions are specific to the uBot-6 mobile manipulator, though it is possible that they could transfer to other robot platforms—especially if their capabilities are similar, such as uBot-5 or uBot-7.

ATG models for ARcubes with eccentric weight distributions differ only in the transition probabilities between aspect nodes as they are designed to be identical apart from their transition dynamics. For example, when flipping, the eccentric weight might result in an almost certain transition to the aspect node with the weight at the bottom.

The actions generate edges in the ATG connecting reachable aspect nodes. For each starting aspect node, Flip and Lift result in four, Push in three, and Orbit in eight different outcome aspect nodes. The transition models for Flip, Lift, and Push are hand-built, while the distributions for Orbit have been learned [152].

Distributions for action parameters are stored along with the transition edges in the ATG and can be sampled when an action is chosen.

5.3 Experiments

We performed experiments in simulation and on a real robotic platform to compare random action selection against action selection under a greedy 1-ply planner in solving an object identification task: the robot is presented with an unknown object and has to identify which model the object corresponds to. It should do so while minimizing the number of used actions.

The simulator is based on the information in the ATG models. This means that the behavior of the simulated environment is guaranteed to match the models. In practice, object models would never be perfect, but only approximations. By using both simulation and real robot experiments, we can also verify if our ATGs sufficiently model the objects. If the models differ too much from the behavior of the real objects, we would see belief condense on an incorrect model/state. Additionally, the simulation is used to demonstrate the scalability of action selection under the greedy planner with increasing model size. To validate the simulation, trends present in these results are compared to those obtained from the real system.

5.3.1 Simulation Experiments – Single Object

The simulator is initialized with a single ATG model and an aspect node. All action outcomes are sampled from the state transitions specified in this ATG. Observations are based on the features expected to be produced by the current (simulated) aspect node.

For each run, a randomly chosen object is selected and simulated for 30 actions. The goal is to identify the object's identity within the model space. We compare the greedy ABP to random action selection by running both of these ap-

Table 5.1. Comparison of random action selection vs. action selection by the ABP ($\alpha = 0.1$) for different model set sizes in simulation.

$\lvert\mathcal{M}\rvert$	**Method**	**Obj. Bel.**	**Entropy**	**Actions**	**Time** (s)
30	Random	0.89 ± 0.24	0.29 ± 0.49	14.2	531.3
	ABP	1.00 ± 0.00	0.00 ± 0.00	4.5	173.8
60	Random	0.94 ± 0.16	0.17 ± 0.22	20.2	756.5
	ABP	1.00 ± 0.00	0.00 ± 0.00	5.1	205.4
120	Random	0.93 ± 0.16	0.25 ± 0.48	18.8	686.1
	ABP	1.00 ± 0.00	0.00 ± 0.00	5.9	264.4

proaches on 30 different simulated objects for each of the three model sets consisting of $|\mathcal{M}| = 30,\ 60,$ and 120. Table 5.1 illustrates the results.

The ABP is used in both cases to incorporate the actions and observations into the belief. Random action selection uses the current belief to find all the actions (type and parametrization) that are possible from the current belief and select one at random. Greedy action selection performs a 1-ply search to determine the action that reduces entropy $H(o_t)$ the most. The entropy $H(o_t)$ listed in the table is the entropy of the population of objects at the end of the trial. The planner takes significantly **fewer actions** and requires **less time overall** in comparison to random action selection. The average time shown describes the time necessary for the object posterior to exceed 0.95 or for the maximum number of actions taken (30). It is important to note that action time dominates; the planning time per action for the greedy ABP was around 1.5 s in the worst case and on average less than 0.5 s. Both approaches theoretically converge to the correct identification as $t \to \infty$, however, these results show only 30 actions or less.

5.3.2 Robot Experiments – Single Object

For the object identification experiments, we present uBot-6 with an unknown object from its model set. Random and greedy action selections are used the same way as they were for the simulation experiments. We performed 10 trials each of the greedy ABP and the random action selection approaches. The results are summarized in Table 5.2. Statistically significant trends indicate that the greedy ABP outperforms random action selection by condensing belief faster (both in time and number of actions taken) towards the correct model. These trends agree with the simulation results for the same task. In Figure 5.4, the slight drops in the correct object belief are a result of action failures; the unlikely outcomes in transitions contribute to increasing belief in other object models that encode such outcomes. The greedy approach has little trouble converging to the correct object, which is similar to the results seen in simulation, but random has more trouble converging to the correct model. Discrepancies between the simulation and robot results can be attributed to

Table 5.2. Single object scene experiment using uBot-6 ($\alpha = 0.1$)

$\lvert\mathcal{M}\rvert$	**Method**	**Obj. Bel**	**Entropy**	**Actions**	**Time** (s)
30	Random	0.48 ± 0.42	0.98 ± 0.72	8.7	1149.1
	ABP	0.98 ± 0.04	0.14 ± 0.21	4.1	680.6

differences in the hand-built transition probabilities and estimated action costs from the actual ones on a real robot.

5.3.2.1 Identification Accuracy

Although entropy in most cases is driven to nearly 0.0, the most important metric for any identification task is accuracy. Table 5.3 shows the accuracy results for the simulated and real robot experiments. The greedy ABP achieved 100% in all tasks, but random action selection misclassified objects in 4 out of 10 trials in the $|\mathcal{M}| = 30$ runs on uBot-6 and 3 out of 30 trials in simulation as well as one in each of $|\mathcal{M}| = 60,\ 120$. We consider a run to have successful classification if the posterior of the correct model has the greatest belief out of all models at the end of the run.

5.3.3 Robot Experiments – Multi-Object Scene

Instead of exposing the robot to just a single unknown object, we present a scene with three unknown objects (Figure 5.5). The robot is tasked to identify all three

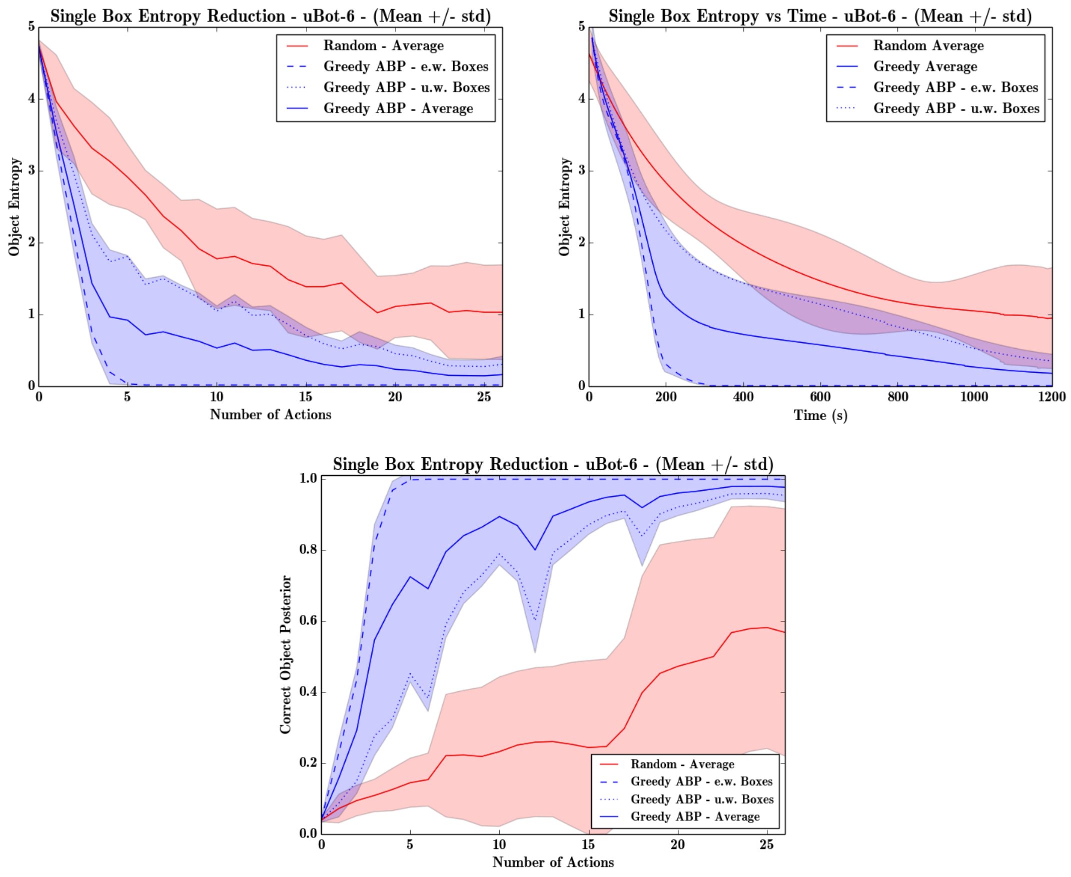


Figure 5.4. Single object identification task with uBot-6 using object model set with $|\mathcal{M}| = 30$. Plots compare the performance of the greedy ABP (blue) versus random action selection (red). Here, 'e.w. Boxes' = eccentrically weighted boxes and 'u.w. Boxes' = uniformly weighted boxes. It can be seen that uniformly weighted boxes are much harder to recognize as their transition dynamics are less distinct than those for eccentrically weighted boxes.

objects with as few actions as possible. It can perform actions on any object in any order. The scene is handled hierarchically in the ABP by evaluating a separate belief for each physical object, evaluating action utility separately, and then choosing the best one (see Section 4.3.2). This keeps the complexity of the planner linear with the number of objects in the scene but denies taking advantage of cumulative benefit one action could have on the belief of several objects. For example, orbiting an object might reveal new features of several objects at the same time. While the planner does not benefit from these interactions, the belief update uses these benefits.

Table 5.3. Identification accuracy computed over all experiments

	$\lvert\mathcal{M}\rvert$	**ABP**	**Random**
Simulation	30	100.00%	90.00%
	60	100.00%	96.67%
	120	100.00%	96.67%
uBot-6	30	100.00%	60.00%

The accuracy of the single object experiments is matched here. The ABP can correctly classify all objects. The beneficial effects of sometimes gaining information on multiple objects at the same time shows in the number of required actions.

Random action selection performs much worse. This is to be expected as the chance of selecting the needed informative action for one object is much lower, because there are three times as many actions to compete with.

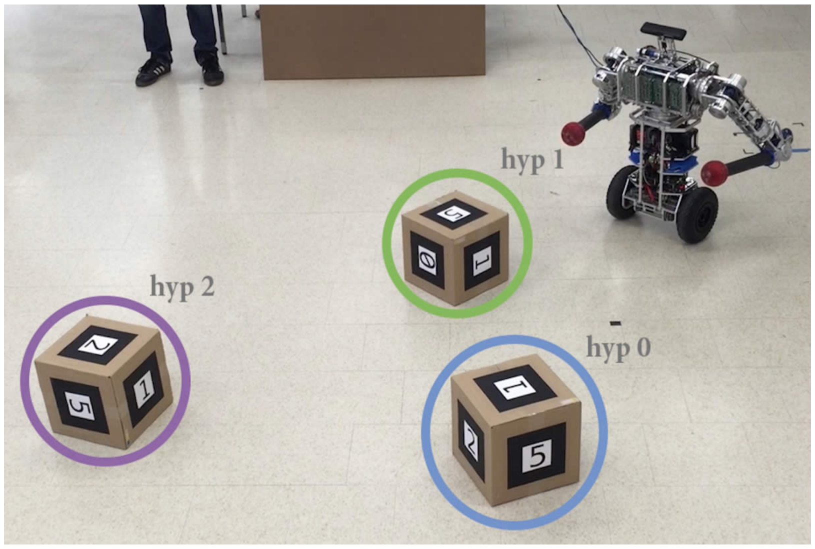


Figure 5.5. Segmentation of a multi-object scene into three hypotheses. The ABP maintains a separate belief distribution for each hypothesis.

5.3.4 Simulation Experiment with PBVI – Orient Task

POMDP solvers do not scale well with increasing problem sizes and usually are intractable for real world robotics problems. Sen used a point-based POMDP solver, heuristic search value iteration (HSVI), to perform ORIENT tasks on boxes though he did not report the required planning times [122]. These experiments used ATG models without geometric information and robot-centric actions. In order to evaluate the performance of such POMDP solvers in combination with our ATG models, we use PBVI to perform several ORIENT tasks on ARcubes.

The planner can be configured with the desired number of belief points and a search horizon. We ran our experiments with a variety of parameter configurations to explore what reasonable parameters are for our problem. We started with a model set consisting of a single ARcube. The corresponding ATG has 48 aspect nodes, resulting in $|X| = |Z| = 48$,x and on average 15 actions have to be considered.

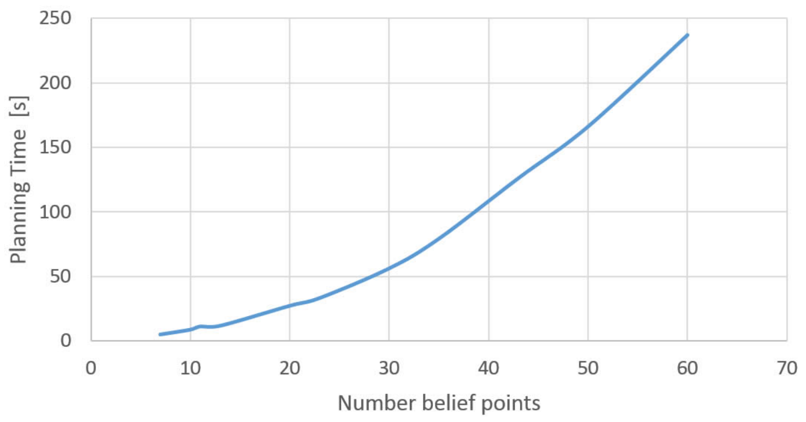


Figure 5.6. Planning time for a single iteration of PBVI over the number of belief points.

For 10 or more belief points, and a horizon of 7, the solver would usually determine good actions that lead directly to the desired task goal. Since the belief points are sampled, sometimes they do not approximate the belief space well, and the chosen action is not one that necessarily leads to the goal. This happens less often the closer the current state is to the task goal and the higher the number of sampled belief points. Nonetheless, the robot still succeeds at the task though at the cost of additional actions since it would recover in one of the following actions. The planning time for an iteration of PBVI is plotted over the size of belief points in Figure 5.6.

The planning time increases linearly with the horizon. PBVI needed around 55 s to plan the next action with 10 belief points and a horizon of 7. Given that this planning time is for a model space of only one object, the planning time will quickly dominate the execution time and become intractable for larger model sets.

5.4 Demonstrations

In order to demonstrate the capabilities of specifying task types for the belief-space planning framework, we use the uBot-6 mobile manipulator (Section 3.1.1) to solve variants of two of the task types described in Section 4.3.3.2: *recognition* and *find*. Assigning different partitions to a task can change the distribution over which the ABP plans, and thus, reconfigure the planner for different tasks. We define task specifications for examples of a *recognize* task and two different *find* tasks and run the planner using their respective task partitions in simulation. This demonstrates the ability of the planner to reduce uncertainty only enough to satisfy the task requirements and be otherwise insensitive. Section 5.4.3 shows how the aforementioned task types can be sequenced for the robot to solve a copying task.

For all these demonstrations we use ATG models for 30 ARcube objects. This model set contains 16 visually unique cubes. Some of these cubes have up to six eccentrically weighted counterparts. For each case, we provide rollouts of the belief over the subsets c_i of partition C. Figures include the belief over objects o_i to illustrate how the subsets of C are composed. Each o_i is colored based on the subset of C to which it belongs.

The 30 ATG models result in a state space of $|X| = 1440$ aspect nodes. At each timestep, an average of $|A| = 15$ actions (action types $\times$ parametrizations) are available. In an artificial data set, aspects on different objects might be identical as they are generated the same way, but on real objects these models would be learned, and with sensor noise, two aspects are very unlikely to be identical. Therefore, we assume the number of possible observations equals the number of aspect nodes $|Z| = |X| = 1440$. Planning time for a greedy action selection takes less than one second.

5.4.1 Task Type: Recognition – "Identify an object"

For the recognition task, we present the robot with an unknown object. Initially, the belief is uniformly distributed over all known object types (or their respective aspect nodes). After the initial observation, the robot repeatedly uses the ABP to select next best actions to execute until the object has been identified. In Figure 5.7, the belief over the object identity can be seen for several time steps until the object is correctly identified as o_{24}. The aspect nodes of each object form a separate subset c_i, and therefore the belief over c is equal to the belief over the corresponding objects o_i. It took five actions for the belief to condense completely on one object.

5.4.2 Task Type: Find – "Find a suitable object"

We demonstrate two examples of the *find* task to showcase two common scenarios. The first task is to *find* an object matching ATG model o_{24}. The robot is presented with an unknown object and needs to determine if this object is indeed object o_{24}. The rollout of the beliefs over subsets c and the object identity o can be seen in Figure 5.8. The color indicates the subset c to which the aspect nodes of each object belongs. Here, the aspect nodes of o_{24} belong to c_1 (blue), while all other aspect nodes belong to c_0 (green).

The robot is presented with the same object as for the *recognition* example in Figure 5.7. The planner chooses a different sequence of actions since it can focus on o_{24} without having to worry about telling it apart from all the other object models, resulting in fewer actions to reach task completion. It only needs three actions to be certain that the object indeed matches model o_{24}.

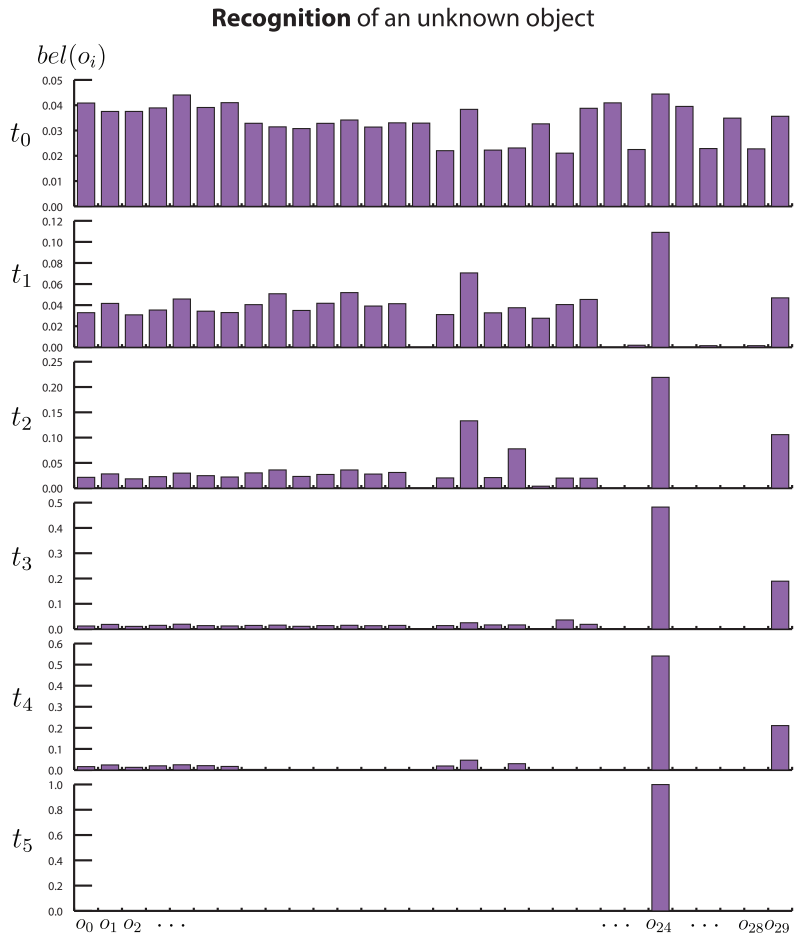


Figure 5.7. Example task to *recognize* an unknown object: a condensed rollout of the belief over object models o_i is shown. The belief over c is equal to the belief over the corresponding objects o_i. Therefore, we refrain from coloring o_i based on their membership in c for readability.

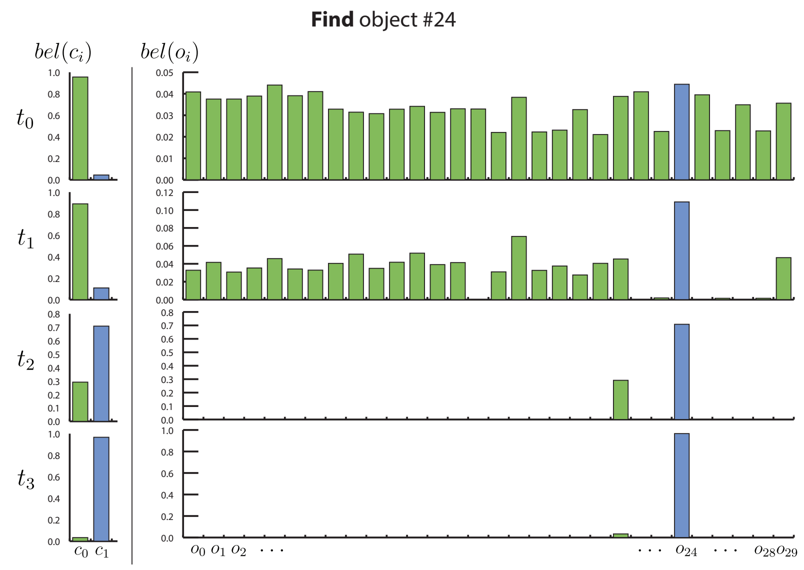


Figure 5.8. Example task to *find* an object matching a specific ATG model (o_{24}) provided as the task specification: a rollout of belief over subsets c is on the left. To visualize how subsets c are composed, the belief over objects o is shown on the right. Target subset c_1 contains all aspect nodes of object o_{24} (blue); those of all other objects are in c_0 (green).

The second task is to *find* an object that could be oriented such that a set of features is in the correct relative position to the robot. In this example, ARtag '1' should face the robot, '4' should be on top, and '2' should be on the bottom of the cube facing the floor. The identity of the object is not important and neither is the configuration the object is actually in. To complete the task the robot just needs to be sure that this is an object that it *could* manipulate to satisfy the specified constraints. The subsets c defining the task can be seen in Figure 5.9 together with rollouts of the beliefs over c and o. The task succeeds without the belief condensing over the identity of the object at hand; the robot can focus on what matters for the task.

5.4.3 Sequencing Find and Orient Tasks for Structure Copying

In this setup, uBot-6 is presented an assembly consisting of two ARcubes. The robot is tasked to observe the target objects and reproduce the structure in a staging

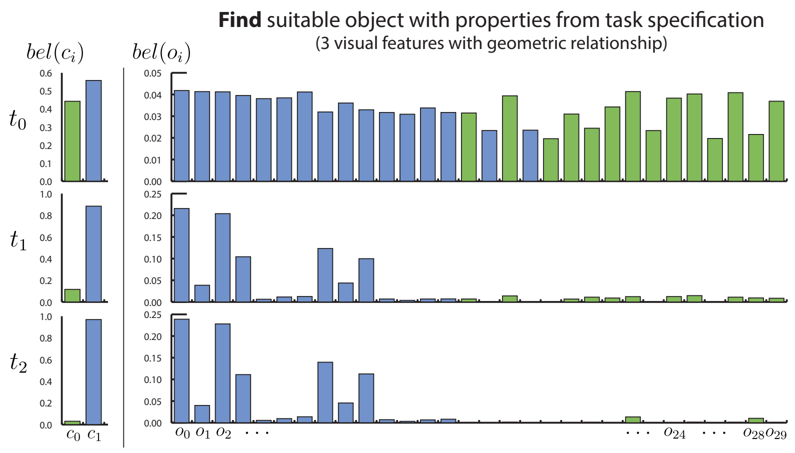


Figure 5.9. Example task to *find* an object with matching features: a rollout of belief over subsets c is on the left. To visualize how subsets c are composed, the belief over objects o is shown on the right. Target subset c_1 contains all aspect nodes of objects that contain the necessary features and can be oriented to expose them (blue); those of all other objects are in c_0 (green).

area (Figure 5.10). Both the original assembly and staging area for the copied structure are known to the robot and contain visual markers on the wall as pose guidance fiducials. For simplicity, the task specification is only based on observations from a single vantage point (one aspect). In general, the task can be more complicated and contain constraints that are not observable from a single vantage point. The robot would have to rely solely on its belief to ensure task success. For example, the robot could take observations from different vantage points and interact with the objects in the target assembly to gather more information in order to replicate it more precisely.

For this experiment, the robot needs determine the task specification from the target assembly. It can then check the staging area to determine what parts are missing to complete the task. It needs to *find* suitable objects in the search scene (Figure 5.11). Once a suitable object has been found, the robot needs to *orient* it to the task specification and pick-and-place it in the designated staging area. We use a hand-built finite state machine to sequence task types from Section 4.3.3.2 for

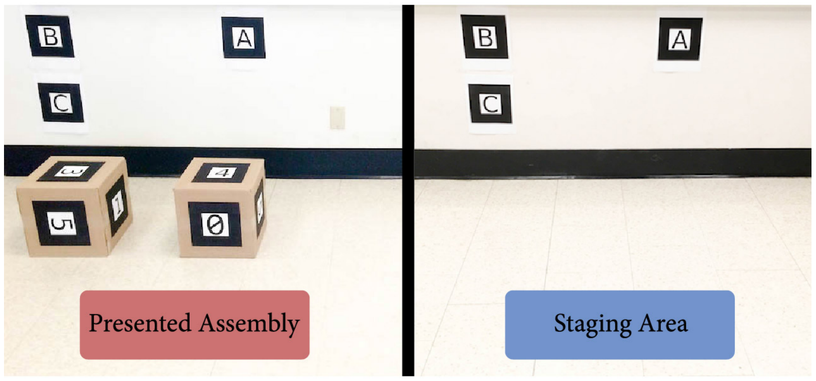


Figure 5.10. Side-by-side comparison of the assembly template (left) and the empty staging area where the assembly should be reproduced by the robot (right). The robot observes the assembly template to get the task specification.

our ABP to complete the structure copy task. The ATG model set used for this demonstration contains 14 object models.

The robot randomly chooses the first object to obtain for pick-and-place. For the situation presented in Figure 5.10, the robot chooses to pick-and-place the right-most object first. To do this, the finite state machine runs a *find* task to locate a suitable ARcube, from the search scene of four ARcubes (Fig. 5.11), that affords an aspect with ARtag '0' in front and ARtag '4' on top ('0-4' aspect). Based on the observation of the assembly template, the robot assigns the partition C following Equations 4.14–4.16: $y(x_i) = 1$ if x_i is a '0-4' aspect. Once the robot is certain that it has found an ARcube with those feature specifications (which takes a single action), it executes an *orient* task to manipulate the cube from the *find* task such that ARtag '4' is on top and ARtag '0' is in front (six actions). The robot assigns the partition C for this *orient* task using Equations 4.16–4.18, where $y(x_j) = 1$ if x_j is a '0-4' aspect. After the robot accomplishes the *orient* task, it uses a pick-and-place controller to grasp, transport, and drop off the cube at the designated location in the staging area.

After placing the first object, the robot goes through the same sequence of tasks for the second object. For the situation presented in Figure 5.10, the robot executes

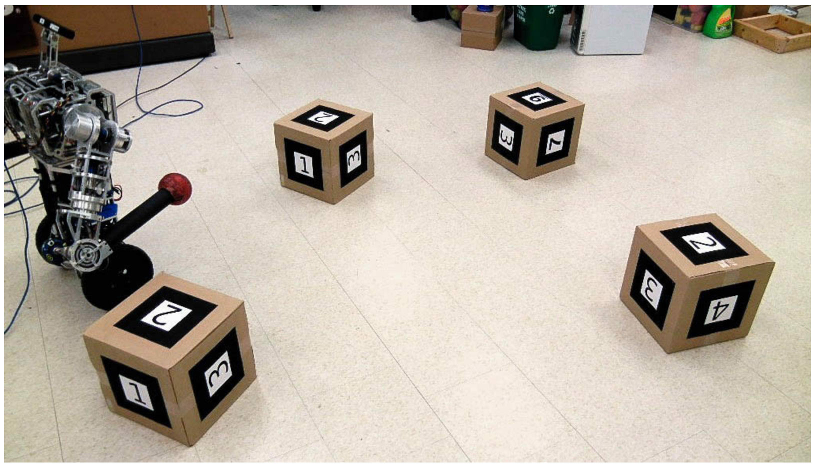


Figure 5.11. Four ARcubes are placed in the search scene. The robot uses a model set of 14 ARcubes. It establishes hypotheses for each of the four ARcubes and plans over them according to the task partitions defined. 11 object models afford the '0-4' aspect and 11 afford the '5-3' aspect.

a *find* task for an ARcube that affords a '5-3' aspect (one action), an *orient* task to reveal the '5-3' aspect (two actions), and a pick-and-place to drop off the cube in the staging area. Figure 5.12 shows the belief over time in the subsets that contain the '0-4' aspect (left) and '5-3' aspect (right). For this demonstration, the execution time heavily dominated the planning time, which was less than 1 s for each action on average. Following the sequence of tasks as presented, the robot was able to successfully reproduce the target assembly as shown in Figure 5.13.

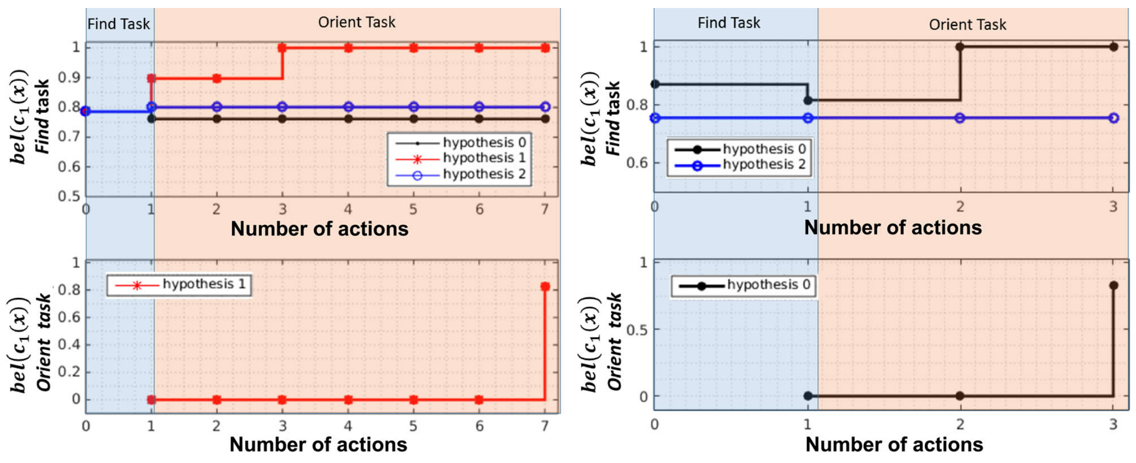


Figure 5.12. These plots show the evolution of belief over the goal subset of the task partition for the objects that were copied (*left*: first object (‘0-4’); *right*: second object (‘5-3’)). The top and bottom plots show the task beliefs for each hypothesis in the scene for the *find* and *orient* tasks, respectively. For the first object, the robot did not register one of the objects in the scene, so it only establishes three hypotheses. The belief for the *orient* task exceeds the threshold after action #7. For the second object, since the robot has already interrogated the scene, it starts with a higher belief prior for the remaining hypotheses. The belief for the *orient* task exceeds the threshold after action #3.

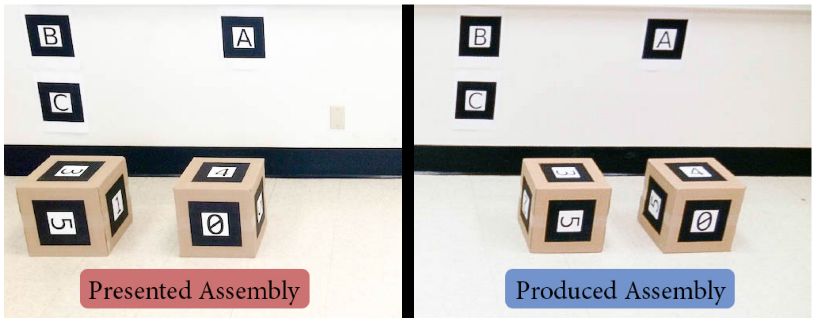


Figure 5.13. Side-by-side comparison of the assembly template (left) and the assembly reproduced by the robot (right). The robot observes the assembly template and copies it in the staging area using objects that it determines to be appropriate from the search scene (Figure 5.11).

5.5 Discussion

Arguably, humans almost never solve tasks optimally. So when evaluating the performance of a robot in a partially observable domain, optimality usually does not matter and is unreasonable to expect. It is much more important that the robot can solve the task robustly despite variance in the environmental constraints and can recover if something goes wrong. Nevertheless, the robot should not stand idle for too long while planning for the next action, and the execution time for actions should usually dominate the planning time.

The presented ABP performed well for all of the tasks we tested in this domain. It completed all tasks successfully and recovered even after a number of highly unlikely action outcomes, such as a box being rotated in the opposite direction due to poorly placed grasp locations. Search depths between 1 and 3 usually provided good enough guidance. For a model set size of 120 objects the search time for a greedy action selection is 0.5 s on average and 1.5 s in the worst case. With adaptive search depths of up to 3, planning times would consistently stay below 5 s. In real world scenarios, it is quite unlikely that belief remains spread over all 120 competing object types as even a single observation can usually condense the belief onto a handful of object types. The ARcube domain was specifically designed such that the ambiguity can remain over many objects even after several interactions and observations. Search times for scenes with multiple objects scaled linearly as expected. The ABP outperformed random action selection in the number of actions, the total planning and execution time, and the accuracy.

The PBVI algorithm was only tested on Orient tasks and could successfully and reliably complete them. The algorithm, without an extension, is not suitable for information-based metrics for many other task types as rewards often depend on the actual belief distribution. The complexity of PBVI is very prohibitive unless the belief has already been condensed to a small number of states. This is usually the case when an Orient needs to be executed. It might be possible to improve the performance of PBVI by optimizing the code. Also heuristicly guided point-based solvers, such as HSVI2 [129] or SARSOP [83], might make results more consistent by avoiding poorly sampled belief points.

The ability to specify different task types for the belief-space planner is very powerful. In the demonstration, we sequenced different task types in a hand-built finite state machine (FSM), but it is possible to use other mechanism, such as symbolic planners, to sequence tasks. This delegates the handling of uncertainty to the belief-space planner at the lower level. Takahashi *et al.* have already demonstrated how this framework can be used to ground symbols for a symbolic planner in belief [139].

CHAPTER 6
DEXTEROUS MOBILITY EXPERIMENTS

uBot-6 and uBot-7 have much extended mobility capabilities compared to typical mobile manipulators. In their primary form of locomotion they dynamically balance on two wheels. Various available postural configurations afford different forms of locomotion such as four-wheeled prone-scooting and knuckle-walking with hands and wheels (Section 3.3). The different postural configurations are reached through postural transitions (Section 3.2.5). These alternatives for locomotion provide great resourcefulness to the robot platforms which enables them to overcome a greater variety of obstacles in mobility tasks. Figure 6.1 shows an example of uBot-6 changing postural configurations in order to get past an obstacle. The added versatility also increases the complexity for planning or reasoning systems. Overall, mobility with several forms of locomotion is refered to as 'dexterous mobility' [80, 115, 116].

In the following sections we first define the dexterous mobility domain. Then we present exploratory experiments that highlight problems for classical path planners when dealing with increasingly diverse action costs that are associated with more resourceful robots. We also show how planning at an abstract level can mitigate these problems. Section 6.3 presents an updated domain for dexterous mobility with partially observable elements. In Section 6.4 we explain how the ATG models can be used to represent environments and can be used as basis for our belief-space planners. Experimental results are presented in Section 6.5.

6.1 Dexterous Mobility Domain

We define the 'dexterous mobility domain' as a navigation domain for robots with diverse action capabilities, such as uBot-6 or uBot-7:

Let W be a description of the known part of the world in the form of one or more occupancy grids. Let P_c be the set of distinct postural configurations of a robot each allowing a different form of mobility. In order to exploit the redundancy in mobility, the robot can re-posture (usually an expensive action) amongst different postural configurations. These transitions as well as motion primitives for each mobility type form the set of actions A.[1] The navigation task is given by start and goal configurations of the robot in Cartesian space together with a postural configuration denoted by n_s and n_g, respectively. Thus, the given domain knowledge, robot capabilities, and task specification are defined by the tuple

$$\langle W, P_c, A, n_s, n_g \rangle . \tag{6.1}$$

A valid solution to such a navigation task is a sequence of actions that lead the robot from starting configuration n_s to the goal configuration n_g. Several valid solutions may exist. The quality of these solutions can be based on different criteria such as execution time or energy consumed. For the purpose of this work, the quality of a solution is based on the total amount of time required to complete the task. Though we assume full observability of the (world) map in this case, once such a technique is deployed on a real robot, state estimation will contain noise, action outcomes will be imperfect, and environments will be dynamic. Therefore, planning is assumed to be tightly interleaved with execution as it might be necessary to produce

[1]The term 'motion primitive' can be somewhat misleading as it has different meanings in the robotics and planning communities. Seen from a robotics perspective, the use of this term implies the fact that the respective motions are the result of complex controllers and motion policies instead of open-loop primitives. In this work, we will use the term in accordance to its use in the planning community as it abstracts the complex behavior as a simple action choice for a planner.

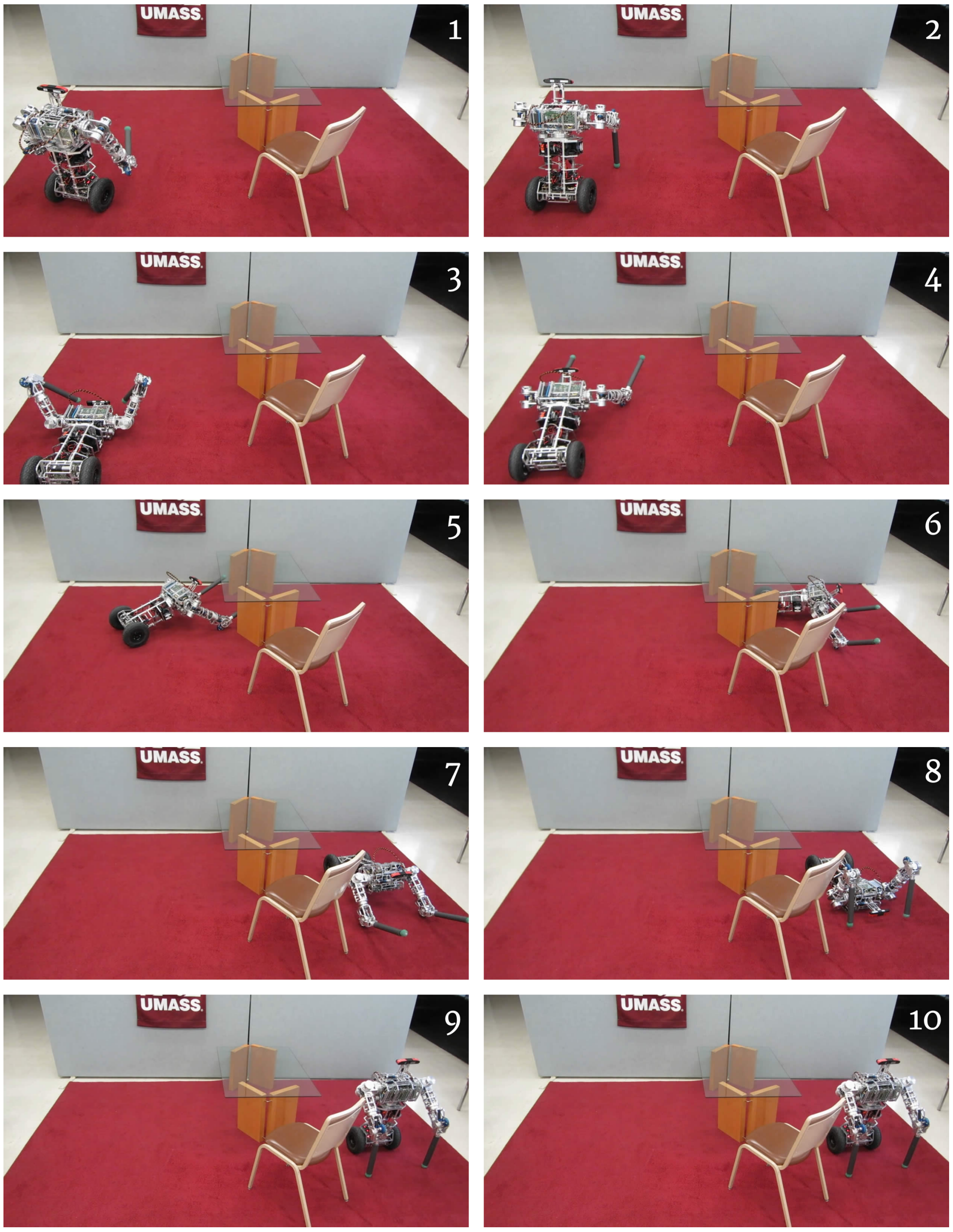


Figure 6.1. An example of uBot-6 changing postural configurations in order to get past an obstacle. It transitions to a 4-point contact configuration, uses its elbow wheels and drive wheels to scoot underneath the table, and transitions back to upright-balancing postural configuration.

updated plans during execution. Thus, the total amount of time required to complete the task includes planning time and execution time of the solution.

6.2 Exploratory Experiments with Classical Planners

There are many planners that are suitable for path planning in standard navigation domains. We perform exploratory experiments to determine how well classical planners for navigation tasks extend to the dexterous mobility domain. Specifically, we are using the uBot-6 mobile manipulator (Section 3.1.1) with a subset of its forms of locomotion. We consider non-holonomic movement with differential steering while balancing upright on two wheels and with Ackermann steering while prone-scooting on four wheels.

The ability to change postural configurations increases flexibility and resourcefulness, but also introduces a significant penalty for changing configurations as transitions require a relatively long time. Heuristic planners will still find optimal solutions, but unless guided by very sophisticated heuristic functions, will avoid expensive actions such as configuration transitions until inexpensive options are explored exhaustively. A*-like approaches using metrics such as power consumption or time are commonly used in state-of-the-art path planning for wheeled robots [90]. Other planning approaches for dexterous robots have been attempted. Notably, motion primitives have been used with probabilistic road-maps [48, 49].

In Section 6.2.3, a heuristic-based planner from the A* family is used to showcase the challenges facing planners that exploit dexterity in a mobility domain. We present a hierarchical variant of A* in Section 6.2.4 and show its impact on planning for dexterous mobility and how it can mitigate problems.

6.2.1 Scenarios

We use three similar example environments of varying difficulty to showcase the utility of dexterity in mobility and also highlight the challenges that arise for planners from it:

Imagine a search and rescue setting. In three scenarios the robot has to move from location A in one room through an angled hallway to location B in another room. In the first scenario (Figure 6.2(a)), the hallway is unobstructed and the robot can stay in balancing postural configuration all the way. In the second scenario (Figure 6.2(b)), both entrance and exit of the hallway are obstructed by an obstacle that prevents the robot from passing through in an upright postural configuration. After changing its postural configuration, the hallway can be traversed by prone-scooting. The third scenario (Figure 6.2(c)) is identical to the second scenario, but the hallway is more narrow. Due to relatively large minimum turn radii while prone-scooting, the corner can only be taken while balancing. Thus the robot has to scoot to enter and exit the hallway but needs to be balancing to take the obstructed corner. For these experiments, the planner has to optimize the time required to execute the planned path.

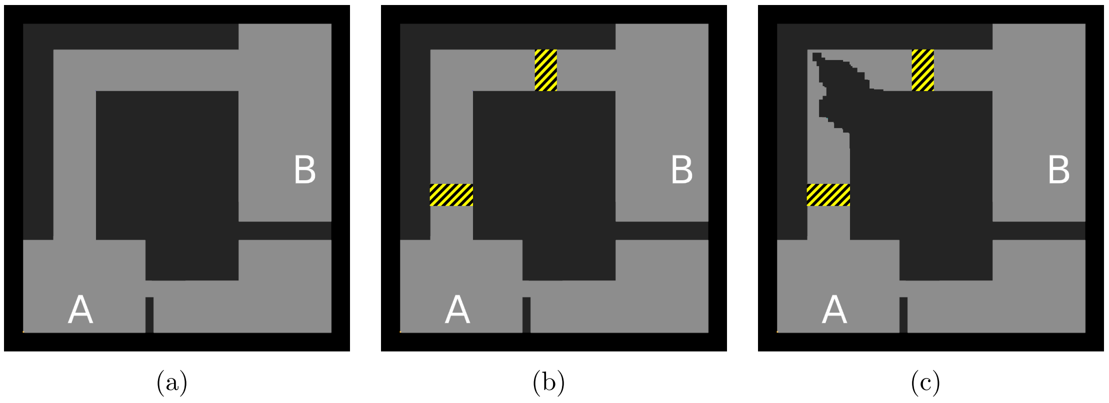


Figure 6.2. Three scenarios for path planning with multiple postural configurations; the robot has to move from A to B. Scenarios 1 – 3 are shown in (a) – (c) respectively. In (b) and (c) the hallway separating A and B is obstructed with low-hanging obstacles (black-yellow shaded areas) preventing balancing mobility. Additionally, in scenario 3, rubble narrows the passage through the hallway, making the section impassable while prone-scooting due to limited turn radii in this mobility form.

The uBot-6 mobile manipulator (see Chapter 3) serves as the experimental platform, but generally any robot capable of several forms of mobility could be used.

Examples of such robots are the iCub which can walk and crawl [94, 34], the Vecna Bear which can switch between balancing and statically stable tracked mobility [111], and the NASA Athlete which has several legs that can be reassigned from mobility to manipulation tasks [147].

6.2.2 Maps

In the dexterous mobility domain the terrain can prohibit the use of specific postural configurations in areas of the map. This could be a low hanging ceiling preventing upright balancing mobility or uneven ground prohibiting anything but knuckle-walking to pass the area.

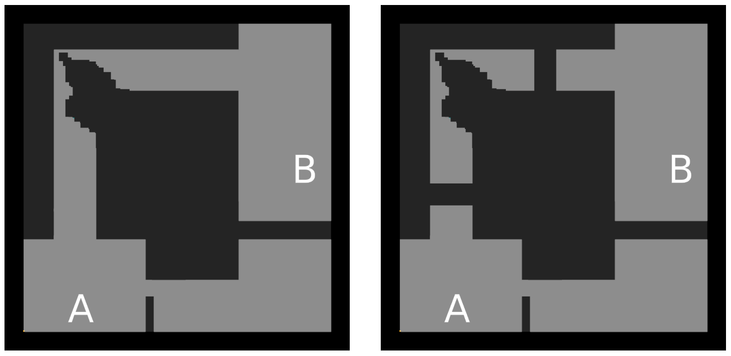
These properties can be embedded in the input map by annotating each location with the postural configurations that are feasible. For our experiments, these annotations were created manually, but in general they can be easily determined from the robot's sensor readings. Sensor readings of the environment indicate if an area affords a specific postural configuration and its mobility form. For example, terrain that affords balancing mobility requires a relatively flat ground and must not be obstructed below the height of the robot.

From an annotated input map we can create occupancy maps for each postural configuration. An example of the obstacle maps for each postural configuration for scenario 3 is shown in Figure 6.3.

For each scenario, obstacle maps are provided for balancing and prone-scooting postural configurations. Both maps can be considered as being dilated such that we can treat the robot as a point.

6.2.3 Classical A*

A standard A* search is used as a representative of classic heuristic search. We refrain from using any of its newer and more advanced variants, such as ARA*,



(a) (b)

Figure 6.3. Obstacle maps for prone-scooting mobility (a) and balancing mobility (b) for scenario 3 (Section 6.2.1).

D*, D* lite, or AD* [92, 134, 68, 91], as we are dealing with fully known static environments in these experiments.

A *state lattice* [105] is used as a discretized representation of the state space. The state space is based on a discretization of the robot configuration. The x, y location is discretized into 4x4 cm grid cells. At each location, the heading θ is discretized into 36 different angles and two postural configurations p_c are available. The connections in the state lattice are generated from feasible actions (see below), and therefore a found path will also be feasible making this representation well suited for path planning. The A* algorithm is used on this state space to find a path to the goal. The action space and the used heuristic are described in the following sections.

6.2.3.1 Action Space

The action space of each lattice state consists of all locomotion and transition actions that are feasible given the current robot pose and configuration. For each postural configuration, a set of actions with different drive distances and turn radii is available (see Figure 6.4).

The minimum turn radius R_{min} varies for the different postural configurations. Upright balancing is the most versatile configuration and allows turning in place

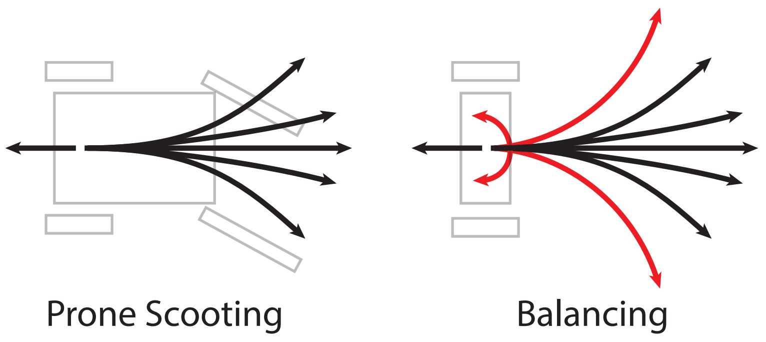


Figure 6.4. Motion primitives for prone-scooting (left) and balancing (right) mobility. Balancing mobility supports turning in place as well as tight turning radii.

through differential steering ($R_{min} = 0\,\text{m}$). Prone-scooting has restrictions on the minimum turn radius with $R_{min} = 1.2\,\text{m}$ for low body height.

Each action results in a trajectory from the current robot pose to another lattice state. An action is added to the action space of a state if no point along the trajectory is in collision with the environment. Depending on the postural configuration, collision checks are performed in different collision maps corresponding to the respective body height and robot footprint. Drive actions are each for a distance of 0.3 m forward and 0.1 m backwards. Rotations in place are by ±0.3 rad. Drive velocities for the experiments are fixed to conservative velocities for each mobility form. Additionally, all transitions to other postural configurations are included if the transition actions can be executed without collision. Costs with respect to execution time for drive and transition actions are shown in Table 6.1.

Table 6.1. Action costs with respect to time for driving and transitioning between postural configurations.

Action	**Cost (Time)**
Drive (0.3 m): Prone-scooting	0.48 s
Drive (0.3 m): Balancing	0.60 s
Rotate in place (0.3 rad): Balancing	0.50 s
Transition: Balancing to low prone	19.00 s
Transition: Low prone to balancing	18.30 s

6.2.3.2 Heuristic

A proven heuristic is extended to explore the problems arising from the dramatic cost differences of actions. The heuristic is based on pre-computed values stored in a *heuristic look-up table* [66], which allows quick retrieval of heuristic values during search. As the entire state space—despite being discretized—is very large, it is not practical to pre-compute and store heuristic values for all states. Instead, similar to the work of Likhachev and Ferguson [90], heuristic values are pre-computed for a simplified state space based on just position and postural configuration (x, y, p_c). This relaxation corresponds to the state space of a holonomic robot. Contrary to the used non-holonomic robot, this relaxed model allows movement in any direction independent of heading. Using Dijkstra's search [37] starting from the goal, all states are labeled with the minimum cost they can be reached with. Movement in 2D uses Euclidean straight-line distance weighted with movement costs in the corresponding postural configuration. Transition costs are the same as those used in the actual search.

6.2.3.3 Results

The solutions for the three scenarios found by the A* algorithm can be seen in Figure 6.5. The blue line indicates the path found and the orange shaded cells illustrate the state space that has been explored during search. A roll-out of the plan found by A* for scenario 3 can be seen in Figure 6.6. Table 6.2 shows the search time and the number of expanded states in each scenario. As expected, scenarios 1 and 2 posed no problem to the A* search as the costs are reflected very well by the heuristic. But in scenario 3, we see a large negative effect when the heuristic is not accurate enough. In this case turning the corner is not possible while prone-scooting due to non-holonomic constraints. As the used heuristic is a holonomic approximation, it considers the area passable and is not accurate enough in this case. As a result all less expensive actions are exhaustively tried before the necessary transitions between postural configurations are considered. It can be seen that both rooms at the bottom have been exhaustively searched. This negative impact increases when the difference in action costs increases or when more cheap alternatives are available. Here, the negative impact of that exhaustive search of less expensive actions was bound by the size of the rooms and would be much worse if more alternatives were available due to larger rooms.

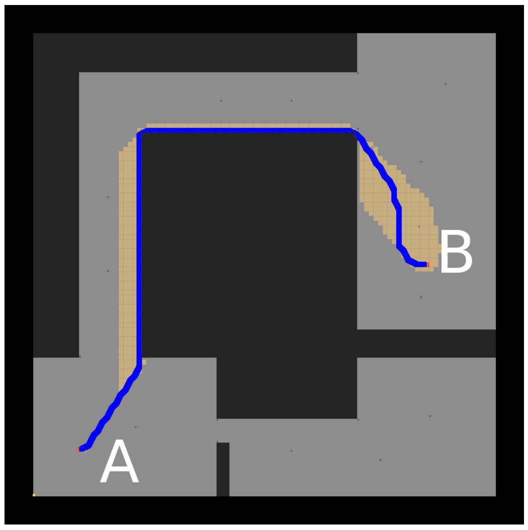
Table 6.2. Performance of A* over the three scenarios

Scenario	**Search time**	**Expanded states**
1	0.21 s	7,327
2	0.51 s	22,850
3	192.73 s	4,798,315

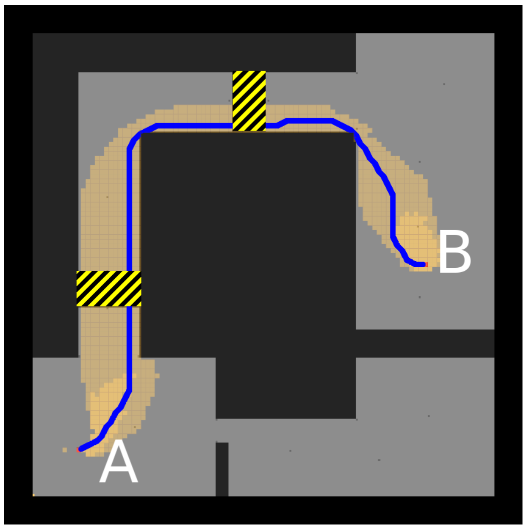
The demonstrated problems arise from heuristic depression regions in the search space where the heuristic values do not correlate well with actual cost-to-goal. The effects are worse with increasing diversity in action costs as a higher number of less expensive actions will be tried exhaustively before necessary higher cost actions are considered. An adequate heuristic can guide the planner to prevent this. But with increasing resourcefulness of robots, the diversity in action costs is likely increasing, and it is also more difficult to create adequate heuristics with increasing complexity.

Dexterous mobility on the uBot-6 introduces only a small number of transitions with diverse action costs, but it already results in very poor planning performance. This problem will become only more prominent with increased complexity of the planning domains arising from more capable robot platforms.

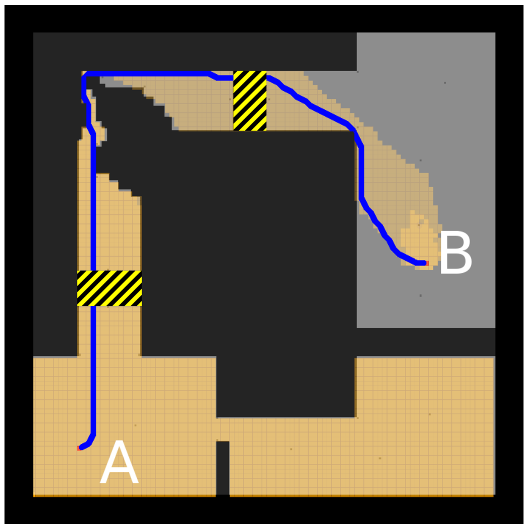
The difficulty of finding suitable heuristics in similar domains has also been noted by Hauser *et al.* [48]. Wilt *et al.* noted the issues stemming from diverse action costs and proposed the usage of distance-to-go as part of the search heuristic instead of or together with the cost-to-go to find solutions in the presence of diverse action costs quickly [119, 148, 149]. In this approach the number of steps to the goal replaces



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

Figure 6.5. Solutions for scenarios 1 – 3: the paths found are shown in blue. Cells shaded orange were expanded in search.

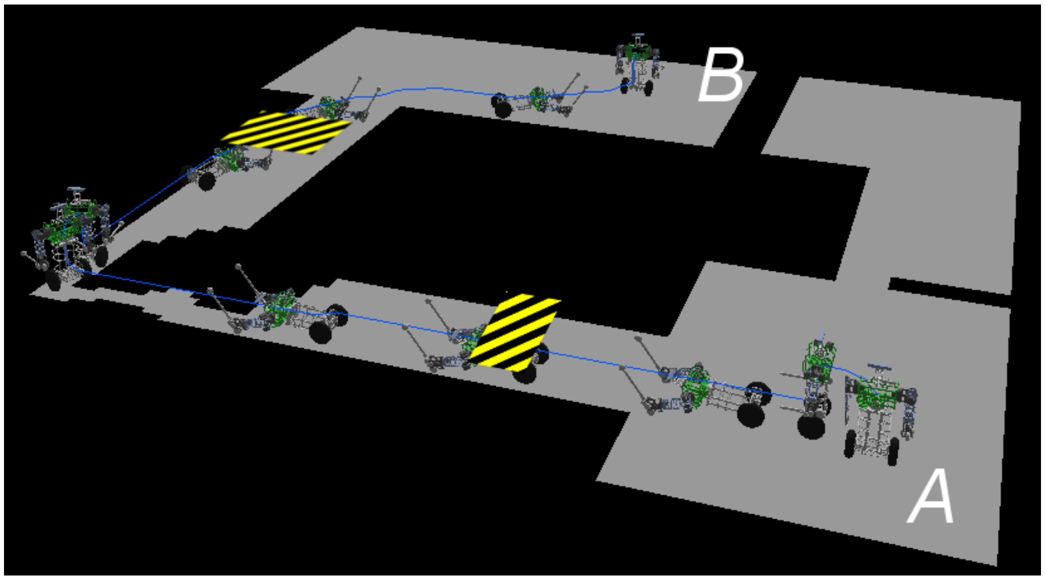


Figure 6.6. Roll-out of the solution for scenario 3 shown in Figure 6.5(c).

the estimated cost to the goal. This is equivalent to unit costs for all actions. As a result, a decision has to be made ahead of time about what trade-off between search speed and solution quality is desired. As an alternative, Aine *et al.* proposed the use of multiple heuristics, each with its own queue for the best states to expand next, to handle heuristic depression regions [1]. This approach indeed enables overcoming situations in which a single heuristic has one depression region, but still fails if there are multiple successive depression regions for the same heuristic. In the following section, we propose a hierarchical approach to mitigate these problems.

6.2.4 Hierarchical A*

As a second search algorithm, a hierarchical planner, denoted as HA*, that employs A* on both an abstract level and low-levels is used. The global planner finds an abstract path in a sparse set of subgoals derived from features detected in the environment which indicate utility for the robot. Paths between neighboring subgoals remain in the same postural configurations. Without the diverse action costs from postural transitions, path planning for these paths does not exhibit the problems encountered in the previous section. Thus, local path planning can be handled efficiently with the classical A* method (Section 6.2.3).

In contrast to existing A*-based hierarchical planners like HPA* [17], the abstract version of the search space is not just a grid with reduced resolution, but is instead dependent on features in the environment which are linked both to exploration and constraints on postural configurations. Subgoal generation is detailed in Section 6.2.4.1.

Once subgoals are generated, A* is run on the resulting subgoal goal graph. Whenever a connection between two subgoals is considered, the classical A* planner (Section 6.2.3) is used as a low-level planner to establish the availability of the action and to calculate the cost. The low-level plans do not contain any postural transitions and thus avoid the problems of diverse action costs. Additionally, the low-level plans have a guaranteed short search time if a path exists and are aborted otherwise.

The sequence of subgoals provides an abstract path to the goal. Unlike the original plan for A*, no optimality guarantees can be given due to the abstraction. But once the required postural transitions are determined, we can skip all other subgoals and use A* to find the optimal path to connect start and end points of postural transitions directly. The existence of the abstract path guarantees the existence of a direct path and also provides an upper bound to the cost. As a result, this plan refinement step is guaranteed to be fast.

6.2.4.1 Subgoals

Our goal is to select subgoals such that the low-level planning for paths between subgoals can ignore postural transitions. Therefore, we determine locations in which transitions between postural configurations are typically useful. For example, a robot can scoot underneath a table. Either side of the table would be a good location to switch between balancing and prone-scooting (Figure 6.7 – left). In general, locations near the boundary of regions with different sets of available postural configurations can be used. Additionally, locations that promise high information gain, for example through visibility around a corner, are of high utility (Figure 6.7 – right). Subgoals can be selected such that the robot's sensor can determine at runtime if the path to the neighboring subgoals is free. Though high information gain or visibility is only useful for partially observable environments, it still offers a good indicator for subgoals in our scenarios.

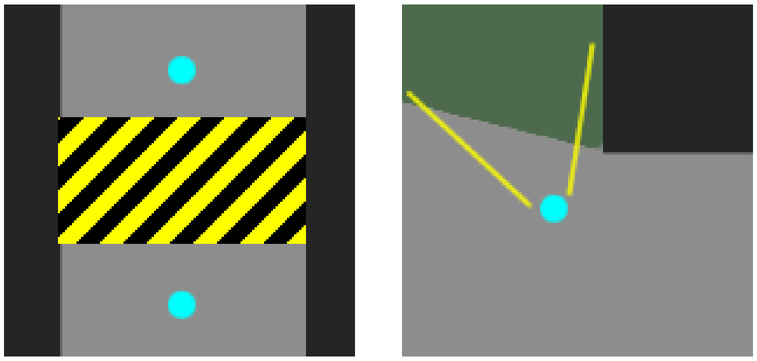


Figure 6.7. Examples for locations (cyan) that suggest utility for postural transitions (left) and high information gain through visibility around a corner (right). Black-yellow shaded regions indicate areas in which no balancing mobility is available. Green shaded area indicates unexplored area.

For the experiments here, we manually selected locations near the boundary of regions with different sets of available postural configurations and based on theoretically high information gain, such as corners or areas without any other subgoals (Figure 6.8). Subgoals are sampled around these locations. These subgoals are located such that a robot can determine visually if the path to a neighboring subgoal is unobstructed.

Both the terrain classification with regards to the usability for postural configurations as well as the detection of subgoals based off of possible information gain could be done automatically based on learned models. Various environmental features, such as volumetric edges with discontinuities in depth as well as the frontier of explored

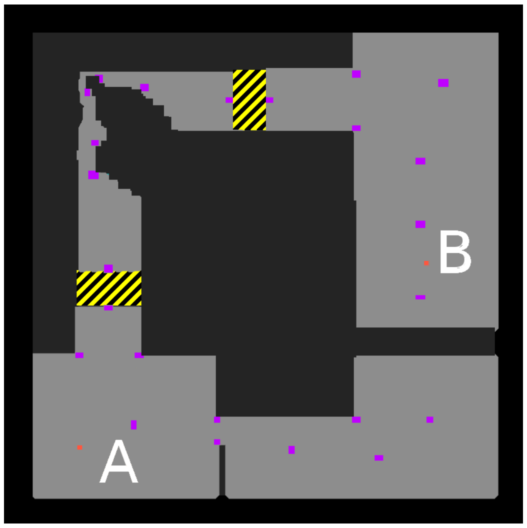


Figure 6.8. Locations for subgoal placement (purple) in scenarios 3: near the boundary of regions with different sets of available postural configurations, based on theoretically high information gain at corners and based on proximity to area without a subgoal.

space can be easily detected. While vertical edges (corners) and the frontier of unexplored territory promise discovery of new features through changed viewpoints in 2D (information gain), horizontal edges indicate places in which the transition into a different postural configuration can both reveal new features and potentially indicate changed terrain constraints.

6.2.4.2 Subgoal Map

From each subgoal only subgoals within the visible range are considered as successors. The reachability and cost of movement between these neighboring subgoals are determined by a local planner. We use the described A* with state lattice (Section 6.2.3) as the local planner. As subgoals are selected such that a robot can visually determine if a path is free, paths between subgoals can be found very quickly even with a simple heuristic based on straight-line distance. As these paths are expected to be very short the low-level search is bound to a maximum search cost dependent on the straight-line heuristic. This prevents long searches when no path is available.

For local searches between subgoals, no postural transitions are allowed, and thus the search is not slowed by action cost differences. At each subgoal, pure postural transitions are considered. The resulting graph based on connections between neighboring subgoals can be considered a topological map.

6.2.4.3 Results

For each scenario, obstacle maps are provided for balancing and prone-scooting postural configurations. The maps were used to manually generate subgoals. HA* was used to solve scenarios 1 – 3. Table 6.3 shows the search time and the number of expanded states for HA* in each scenario. Figure 6.9 shows the solutions found by HA* for scenario 3. The search time for scenarios 1 and 2 increased slightly, but remained in a tolerable range. By reducing the state space through the use of subgoals informed from environmental features, the hierarchical planner yields a substantial improvement in the search time in scenario 3 from 192.73 s down to 13.01 s though the found paths are not optimal any more.

Table 6.3. Performance of hierarchical A* with sparse subgoals (HA*) over the three scenarios

Scenario	Search time	Expanded states
1	1.82 s	28
2	6.54 s	73
3	13.01 s	144

6.2.5 Discussion

We presented several exploratory experiments in simulation that are focused on the effects of dexterity in mobility. Two algorithms were used to solve tasks in the dexterous mobility domain. Navigation tasks have to be completed while overcoming environmental constraints by choosing appropriately from functionally equivalent actions and action sequences that vary in costs and availability.

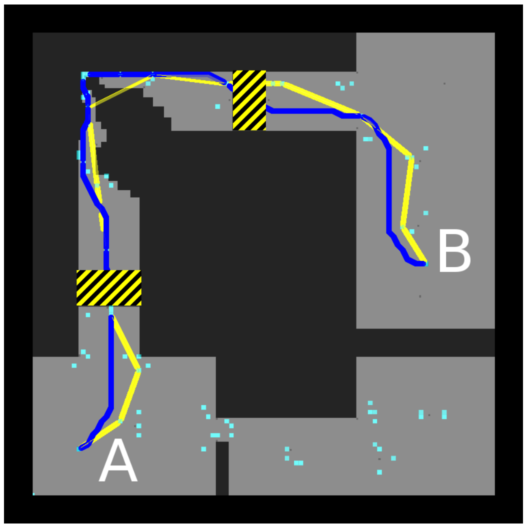


Figure 6.9. Solution of the hierachical A* variant for scenario 3. The sampled subgoals are shown in cyan. The abstract path found is shown as a sequence of subgoals in yellow. The refined solution is shown in blue.

A state-of-the-art A*-based planner was used and revealed challenges in the presence of diverse action costs. The demonstrated problems arise from heuristic depression regions in the search space where the heuristic values do not correlate well with actual cost-to-goal. The effects are worse with increasing diversity in action costs as a higher number of less expensive actions will be tried exhaustively before necessary higher cost actions are considered. An adequate heuristic can guide the planner to prevent this. With increasing resourcefulness of robots, the diversity in action costs is likely increasing, and it is more difficult to create adequate heuristics.

We proposed HA* as second algorithm that uses A* hierarchically on subsections of the task. This algorithm was impacted less by the diverse action costs. For these experiments, potential subgoals were manually created, but generating them from the environment using models and sensor is feasible. The quality of subgoals is important for the performance of this algorithm, and we propose to generate them based on the utility of parts of the environment with respect to the robot's skills.

In the next section we extend the dexterous mobility domain with partial observability. Then we show how the ATG models (Section 4.2.1) can be used to model an environment on an abstraction level relevant to the robot. This method uses the idea of planning on an abstracted level from HA* (Section 6.2.4) to overcome issues arising from diverse action costs. Additionally, this environment model can handle partially observable elements of the environment and uncertainty in actions and observations.

6.3 Partially Observable Dexterous Mobility Domain

In this section, we extend the dexterous mobility domain from Section 6.1 with partially observable elements. This will enable us to model tasks and environments in which a robot has to handle various sources of uncertainty.

Like for the dexterous mobility domain, a map of the world is given, but the pose of the robot in the environment is uncertain as a result of noise in odometry and the uncertainty of actions and observations. This is very similar to a typical navigation domain with partial observability [127, 73].

In the case of dexterous mobility, a robot is usually reluctant to switch forms of mobility as transitions are often very costly. A robot will keep using its current mobility form unless there is a clear benefit in switching. So if a robot encounters an obstacle that it cannot overcome with the current mobility form, it can either switch to a different mobility form or find a way around it. Despite the uncertainty in the robot's pose, this decision can be handled as described in the previous section.

In order to make this domain a bit more interesting, we introduce variable elements, such as doors, to the environment that control the connectivity between regions. Their location is known, but their state is only partially observable. For example, a closed door or a blocked hallway could prevent access to the next room. Now it might be worth to take an information gathering action first to see if a potentially better path around the obstacle is accessible before taking expensive contingency actions. The state of these connections can be easily established visually or haptically for most navigation tasks. An example can be seen in Figure 6.10.

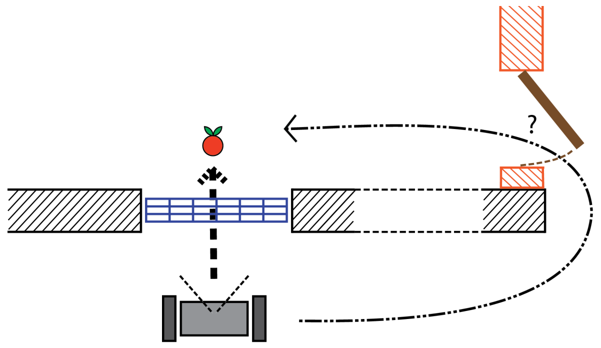


Figure 6.10. Example of the extension of a map with variable, partially observable elements such as doors. The robot has two options to reach the goal: 1) by performing a certain but costly postural transition and scooting underneath an obstacle (blue) or 2) by trying to use the cheaper drive around the corner through the hopefully open door. It might be worth just driving to the corner to determine the state of the door.

We adapt the definition of the dexterous mobility domain (Section 6.1) with minimal changes. We extend the world decription W with probabilities representing priors on the availability of the respective connection. They can be provided as annotations on the occupancy map(s) or as annotations on the edges of a topological map.

6.4 Representation for Belief-Space Planning

In order to use a belief-space planner to solve tasks in this domain, we need to define suitable state, action, and observation spaces. As discussed in Section 4.2, the choice of representation is usually a trade-off between expressiveness and complexity.

Typical navigation domains with partial observability are often based on uncertainty in the pose of the robot in the environment as a result of noise in odometry and uncertainty of actions as well as visual ambiguity of places (several rooms might look similar) [127, 73]. We find that this uncertainty is handled well through control and state estimation techniques. Action outcomes for locomotion might be uncertain on the lowest level as single motion primitives are executed slightly imperfectly and odometry is noisy, but once put together in closed-loop controllers and sequences of actions, these uncertainties are detected by state estimation and corrections are integrated in actions at the next time step. The challenges in the state estimation can be handled quite well with approaches such as simultaneous localization and mapping (SLAM) [39, 141]. Therefore, neither partial observability in the precise pose of the robot nor the uncertainty in action outcomes are important enough to justify increasing the complexity for planners.

In the context of dexterous mobility, we are more interested in taking appropriate higher-level decisions on which mobility form to use rather than determining the precise path to the goal. The most important information for such a decision is which mobility forms are available and are they afforded by the environment. This can be represented well with topological maps of regions of the environment.

Often topological maps of the environment are used for path planning instead of using the precise pose of the robot as state for the planner [18, 81, 72, 127]. Togological nodes represent regions of robot state space, with pose and postural configuration, that can be grouped as they provide the same information of the environment and afford the same actions. Edges represent actions that bring the robot from one node to another. The connections between topological nodes are short and can be handled with ease by path planning and execution. Instead of uncertainty over the precise pose of the robot, we can maintain a belief over the location of the robot in a topological map [23].

This aligns well with the results from Section 6.2 where we presented how planning on an abstracted level, such as a topological map, can mitigate problems diverse action costs. The ATG models defined in Section 4.2.1 can support all the needed functionality to completely represent this domain for our belief-space planners.

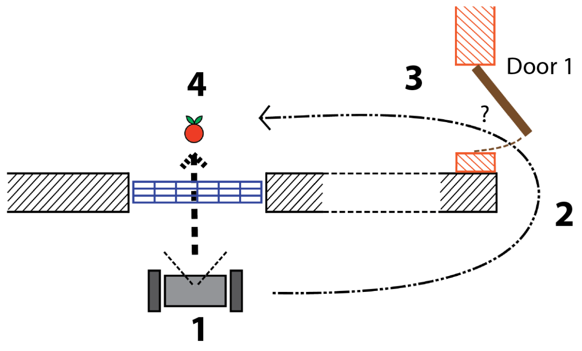
6.4.1 Environment Model

The ATG models presented in Section 4.2.1 are based on a multi-graph connecting abstracted states of the robot with respect to an object through actions. In a navigation task, the state of the robot is with respect to the environment instead of an object. An ATG for an environment can be considered similar to a topological map. Aspect nodes represent abstract states of the robot with respect to the environment, e.g. 'in room 1', and edges represent actions that support moving from state to state.

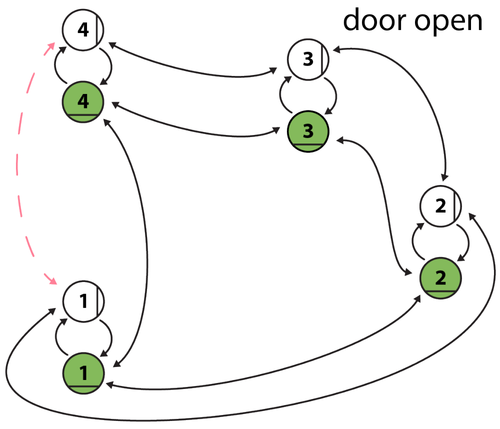
In comparison to aspect nodes for objects, usually a much less detailed representation is required to generate a meaningful map of interaction capabilities. The aspect nodes are still based on aspects of perceivable, salient features. Aspect nodes can be created directly from a topological map or from occupancy grids with appropriate annotations, much like we created subgoals in Section 6.2.4.1. We assume that for every variable connection the associated aspect nodes support observation of the connection availability. These aspect nodes will provide a good coverage of the environment. By including the postural configuration as part of the set of perceivable features, postural configurations are seamlessly integrated in the state estimation of our belief-space planners. As a result, for each place we create a separate aspect node for each postural configuration. They are connected by edges representing the postural transitions.

Unlike the approach used for HA* (Section 6.2.4), the edges in the ATG already contain cost estimates for the associated actions. This removes the need to determine a cost estimate through a low-level planner. Actions on the edges of the ATG, such as mobility and postural transition actions, are usually reliable unless catastrophic failures are considered. Therefore, we consider transitions to be deterministic for these experiments, though this is not a technical limitation.

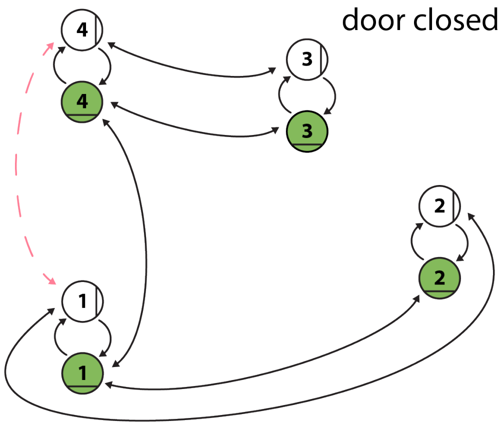
A distribution over aspect nodes $bel(x)$ represents the belief state of the robot. In the ATG described so far, a belief distribution that is not condensed on a single aspect node can only occur when the history of actions and observations supports multiple possible states of the robot, for example by similar looking rooms.



(a)



(b)



(c)

Figure 6.11. Example of the ATGs representing the map for different states of a variable element in the map: door 1 open (b) and door 1 closed (c). Node numbers in the ATGs correspond to the places in (a). Green nodes represent a prone scooting postural configuration, white nodes a balancing configuration. The red edge from 1 to 4 shows the unavailable passage while balancing. If the state of the door is unknown, then current belief is evenly distributed on node ‘balancing 1’ in both ATGs. All four aspect nodes for place 4 are valid goal states.

To represent the belief over variable, partially observable elements in the environment, we generate separate ATGs for each variation of the environment. For example, for the environment shown in Figure 6.11(a), we have an ATG for an environment with *door 1 open* (Figure 6.11(b)) and an almost identical copy with *door 1 closed* (Figure 6.11(c)). The ATGs differ in the perceivable features for the open or closed door and missing edges that represent driving through the open door in both mobility modes. If the robot cannot influence these variable elements, then no transitions exist between the aspect nodes of these ATGs. In the future, they could be easily extended to include edges for opening and closing doors. Now, if the robot is certain about its own location, all belief will be condensed on a single aspect node of each ATG since the state of variable elements remains uncertain. Every time the previously unknown state of a variable element is observed, the number of ATGs with non-zero belief will be halved.

6.4.2 Task Representation

A navigation task in this representation can be expressed as an *orient* task (see Section 4.3.3.2.4). When dealing with objects, the robot can either reposition itself with respect to the object, or manipulate the object with respect to itself to complete the task. In the case of the environment, the robot can reposition itself with respect to the environment or possibly alter variable elements in it. For a simple navigation task our task partition consists of two subsets:

$$c_1 = \{x_i | y(x_i) = 1\}, \tag{6.2}$$

$$c_0 = X \setminus c_1, \tag{6.3}$$

where $y(x) = 1$ iff the aspect node x represents the desired goal location and the desired postural configuration. If we do not care about the variable elements of the environment, then this would select one or more aspect nodes for each ATG.

This representation enables a planner to search for a task solution on a level of abstraction. This abstraction avoids the problems stemming from diverse action costs and heuristic depression regions. The ATGs model cost estimates for all actions, so low-level planning only has to be used when executing an action, thus improving planning time. The representation supports maintaining a belief distribution to handle ambiguous areas in an environment and also handles variable, partially observable elements. Though the number of ATGs grows quickly with the number of variable elements, observations also quickly reduce the number of ATGs with non-zero belief.

6.5 Experiments

We are evaluating our ATG in combination with HA*, PBVI, and the belief-space planner on a small dexterous mobility task with partially observable elements in the environment. Due to the high diversity in the action costs of postural transitions and mobility actions for uBot-6, it is relatively difficult to create situations in which overcoming an obstacle with postural transitions and an alternative detour are close in cost. The robot can easily drive 20 m in the same time it can transition into another postural configuration and back. Therefore, creating realistic scenarios for the real robot requires a large space and related control infrastructure. Instead, we chose to evaluate the planners in simulation.

We provide topological maps as input to the planning framework. The map is then turned into several ATG models. Each permutation of the states of the variable elements in the environment is modeled in one ATG. We use scenario 3 from Figure 6.2(c) and add a door with unknown state to connect the rooms on the right. The topological map contains 13 subgoals. Edges are labeled according to traversability in each postural configuration. This map results in two ATGs, one for the open and one for the closed door, with 26 aspect nodes each and about 70 edges.

6.5.1 Results

We tested the hierarchical A* (HA*) algorithm in conjunction with the ATG models. Instead of low-level A* planning interleaved with the high-level A* plan, the connectivity and edge costs are retrieved directly out of the ATG models. The low-level A* path only needs to be calculated when the corresponding transition (ATG edge) is being executed. The robot only worries about the details once it needs to. This is very similar to the concept of the hierarchical planning in the now (HPN) framework [60]. Paths for all of our maps were found in less than 10 ms.

The belief-space planner and PBVI do not try to find a full path to the goal but only selects the best action for the current belief state. For each state during simulation, our active belief planner successfully selected the correct actions to reach the goal in less than 1 s. Despite a high search depth being required, the belief is highly condensed and the effective branching factor in the belief rollout is very low.

PBVI selected the correct action only sometimes and even then required much longer time (20 s and up). The inconsistency can be attributed to the sampling of belief points. As the path is along a long chain of aspect nodes, a poorly sampled set of belief points can result in a poor propagation of the reward from the goal. It is possible that a heuristically guided version of point-based value iteration, such as HSVI2 [129], could be faster and more consistent here. Even for this small map, PBVI shows poor performance and is likely not tractable for larger maps.

6.6 Discussion

With uBot-6 we introduced several postural configurations and associated mobility modes. While obstacles in the environment can prevent passage for a single mobility form, the robot might be able to switch to another postural configuration and overcome the obstacle with the associated mobility mode. We introduced a path planning domain for robots with such versatile mobility. In our experiment, we could highlight the problems that arise for classical planning methods when planning in this domain. The transitions between postural configurations introduce high diversity in action costs and can lead to heuristic depression regions if the heuristic does not approximate the capabilities of the robot close enough. In heuristic depression regions, planners are reluctant to consider expensive actions until less expensive actions and sequences of actions are exhaustively explored. This can often lead to intolerable planning times.

We contend that robots have very reliable basic capabilities, and this can be used to approach planning hierarchically. The problem can be divided into parts that are individually very easy to handle for the robot. The planning can then happen on this abstracted level without the need to deal with the details that are handled robustly through established control and low-level planning.

We have shown how a hierarchical version of A* can handle the problems in our experiments. This planner used A* on two levels such that a full low-level plan was found at the end of planning. In conjunction with our ATG models, a hierarchical A* planner can rely on modeled cost and connectivity throughout the map to quickly find a path and only perform a low-level A* path plan when a path segment is about to be traversed. This concept is very similar to the hierarchical planning in the now (HPN) framework [60].

The ABP can be used as well for planning in this dexterous mobility domain with partially observable elements though it needs to be evaluated further on larger maps as larger required search depths could become prohibitive. Additional information-based metric could be used in combination with heuristics to address this potential problem. PBVI was very slow on our small sample problem, and the action selection was not consistently successful at leading to the goal. This can be attributed to sampling issues similar to the results in Chapter 5.

CHAPTER 7
FUTURE WORK

The robot platforms, models, and the planning framework introduced in this thesis can be extended in several important directions. Some possible extensions have already been discussed in each individual chapter, and we highlight additional future research directions:

7.1 Articulated Hands for Improved Manipulation Capabilities

The current spherical end effectors are sufficient to interact with buttons, switches, and levers [71], and in our experiments, uBot-6 uses bimanual grasps and single-handed pushes to manipulate objects. The spherical end effectors also proved very useful for postural transitions without having to worry about protecting fragile fingers. But the uBot platform has limited capabilities with respect to one-handed grasping and grasping of smaller objects without articulated hands. Articulated hands could greatly improve the capabilities to manually interact with objects and the environment and would especially support one-handed grasping. Several hand concepts have been investigated that integrate fingers in mechanisms suitable for the unique requirements from postural configurations and transitions. This includes retractable fingers and flexible, underactuated fingers similar to the Yale OpenHand [96]. The final design and integration remain future work.

7.2 uBot-7 Integration – Improved Performance

All of the experiments in this work have been conducted on the uBot-6 platform. Once fully integrated, the uBot-7 should be able to run the same control framework and perform the same tasks. It should have improved capabilities to sense forces and torques while manipulating objects. This will be useful for peg-in-hole style assemblies. The improved head should give it more options to acquire visual observations in all supported forms of mobility without the need to use the base.

7.3 Autonomous ATG Model Learning

ATG object models are currently partially learned and hand-built. It will be very interesting to learn the complete object model autonomously. This includes mechanisms to transfer modeled knowledge between objects with similar properties or between different robots. In the case of the ARcube domain, the properties and interaction capabilities could be learned autonomously on a single ARcube object. When encountering a new ARcube object, the modeling process could be primed with the known similar model. The features would have to be relearned, but the interaction capabilities with action parameters and transition probabilities should be similar. Likewise, for a new robot, model learning could be bootstrapped from already known ATG models of another robot. Modeled feature locations and surface normals could remain the same, but interaction capabilities would depend on the abilities of the new robot and might need to be relearned.

7.4 ATG Models for More Object Types

So far ATGs have been used to model ARcube objects and environments. While the ARcube objects provide a perfect domain to stress planning frameworks, it will be interesting to see how well more general objects can be modeled. A first step would be to model ARcubes without using the ARtag features, but instead relying on basic

2D features and on volumetric features such as edges, corners, and planar faces. This would enable a comparison of the new models with respect to the idealized ARcube version.

7.5 Extension of ATG Models with Tactile Properties and Force/Torque Guided Assemblies

The ATG models currently support state representation and belief condensation based on visual features and on the transition dynamics of actions. Tactile properties and surface normals are promising properties to extend the ATG models. The integration of these properties could greatly improve the versatility of the belief-space planner, making it less reliant on specific sensor modalities. Modeling tactile properties and surface normals can enable prediction of contact points or support surfaces for object-object assemblies. Likewise, modeled force/torque responses can be used to perform peg-in-hole style object-object assemblies.

7.6 Orientation Distributions for Improved Interaction and Matching

Currently, our framework determines the similarity of observed features with aspects solely based on the geometric constellation of features. We used logical constraints to remove support for infeasible aspect nodes. This mechanism could be much improved and generalized by modeling an orientation distribution for the object frame for each aspect node. A suitable method would be the Bingham distribution, which is used to represent uncertainty directly on the unit quaternion hypersphere [15, 43]. Quaternions avoid the degeneracies of other 3D orientation representations, while the Bingham distribution supports modeling of large variance rotational distributions. The modeled orientation distributions would eliminate support for incorrect aspect nodes that are partially supported by the observations, but infeasible given the orientation of their constellation. This would greatly improve state estimation as well as pose estimation. At the same time, transition actions and probabilities such as 'ORBIT' could be informed by the current pose estimate. For example, if the robot is in the center or at the border of the state space region associated with an aspect node, the current pose estimate together with the orientation distribution of a target aspect node can inform the robot if a 10° orbit is sufficient over a 45° orbit.

7.7 Extension to Multi-Task Planning

In the presented framework, the belief-space planner selects actions based on a specified single-object task. This can easily be extended to handle several tasks at the same time. For each task i we provide a partition C_i and a corresponding target distribution T_i. As the definition of a task partition does not alter the belief representation or the distribution of belief, the resulting information-based metric $M_i(c_{i,t}, T_i)$ can be evaluated at almost no additional cost. Given weights w_i, the planner can evaluate the quality of actions from the linear combination of the different tasks:

$$M = \sum_i w_i M_i(c_{i,t}, T_i). \tag{7.1}$$

Action selection would try to find the action that provides the highest utility for all tasks together. Additionally, it is possible to perform logic operations, such as AND or OR, to combine several tasks.

7.8 Fully Recursive Application of the Belief-Space Planner

We have shown how tasks for the ABP can be sequenced to perform tasks involving multiple objects. For this demonstration, the individual steps were sequenced in a hand-built finite-state machine. Instead, the higher level plan could be generated by a symbolic planner or by a recursive application of the belief-space planner. Takahashi *et al.* have already demonstrated how this framework can be used to ground symbols for a symbolic planner in belief [139]. Additionally, preliminary work has been done to use a second belief-space planner to take higher level decisions [85]. A fully recursive application of the same planning framework remains the goal for future work.

CHAPTER 8
SUMMARY AND CONCLUSION

8.1 Summary

The main goal of this thesis is to improve the state-of-the-art for robots performing tasks in unstructured environments. In order to perform well in such environments, a robot needs to be resourceful in its sensory and motor capabilities and also be capable of efficient action selection to exploit this resourcefulness based on task and environment constraints. In this thesis, we addressed this problem from three directions:

First, we presented a new design concept in resourceful robot platforms, called the UMass uBot design concept, and introduced the sixth and seventh in the uBot series. The concept combines many advantageous properties into a single robot platform while remaining easy to use and low-cost. uBot-6 introduces unique access to various postural configurations and the associated forms of mobility. uBot-7 adds series elastic actuators (SEAs) to make the robot more robust and safer around humans.

Secondly, we introduced a powerful extension to an existing affordance-based object model. We used aspect transition graphs (ATGs) to model both objects and environments with respect to the capabilities of the robot. By extending the ATGs with geometric information, we enabled improvements such as easy object pose estimation, robustness to occlusion, better observation models, and more robust actions. As an ATG model is based on perceivable features of an object as well as the interaction possibilities known to the robot, the ATG model can provide abstractions that represent the states of an object that matter to the robot. The use of this abstraction keeps the state space compact without losing expressiveness.

Finally, we presented a belief-space planning framework, called the active belief planner (ABP), that is capable of efficient action selection on a real robotic system under partial observability. The framework uses ATG models to acquire a compact but expressive state space, transition models, observation models, and cost estimates. Additionally, the ATG guides the planner by providing the actions that are known to cause useful interactions with an object. A novel method specifies different single-object tasks for the belief-space planner. It defines a task partition over the state space and combines it with an information-based metric. The expressible granularity of tasks depends on the detail of the ATG models and thus on the capabilities of the robot. We show how this mechanism supports different task types commonly encountered in robotics.

In order to evaluate the performance of the proposed framework, we introduced two challenging, partially observable planning domains for real robots. The *ARcube domain* provides a very challenging benchmarking system that is able to stress planners with high degrees of ambiguity in multiple sensor modalities and requires manual interactions with objects. The *dexterous mobility domain* defines navigation tasks for robots with the ability to use several forms of mobility. These domains are used to evaluate our planning framework in simulation and on the uBot-6 robot.

8.2 Conclusion

Arguably, humans almost never solve tasks optimally. So when evaluating the performance of a robot in a partially observable domain, optimality is usually not important and is unreasonable to expect. It is much more important that the robot can solve the task robustly despite variance in the environmental constraints and can recover if something goes wrong. Nevertheless, the robot should not stand idle for too long while planning for the next action, and the execution time for actions should usually dominate the planning time.

The presented ABP performed well for all tested tasks in both domains and could reliably solve all tasks. Even in recognition tasks in the highly ambiguous ARcube domain, the robot succeeded consistently and recovered even after a number of highly unlikely action outcomes. Myopic planning to a horizon between 1 and 3 usually provided good enough guidance to find good solutions to the tasks and required only very short planning times.

Generally, belief-space planning can handle partial observability and uncertainty in action outcomes and observations. In practice, there are still very few robot systems in which belief-space planning is used to deal with more than just limited special purpose tasks. This is mainly because of poor scalability with respect to state, action, and observation spaces. Additionally, planners often rely on expensive simulation at runtime to estimate transition and observation models. We have demonstrated that the methods introduced in this thesis enable a real robot to robustly solve various task types in challenging domains, and that they can do so quickly. This framework has the potential to be generalized to a wider variety of tasks and domains but much still remains to be done.

We observe that a key ingredient for the good performance is the choice of knowledge representation. Our framework uses the ATG model which can represent objects and environments alike and combines several strong advantages. As an ATG model is based on perceivable features of an object as well as the interaction possibilities known to the robot, the ATG model can provide abstractions that represent the states of an object that matter to the robot. For an ideally built or learned ATG model, the resulting abstract state space is as compact as possible without losing expressiveness. This is important as the complexity of planners heavily depends on the sizes of the state and observation spaces. The impact of the choice of this state space is also visible in the proposed task representation. The task representation works well with the ATG-based state representation as this ensures that tasks are again specified on a level that is relevant to the robot and thus enabling very expressive task definitions.

In addition to the good trade-off between compactness and expressiveness, the ATG model provides transition and observation models and thus eliminates the need to use a simulator to acquire these models at runtime and enables more efficient planning. The ATG model also suggests useful known interactions that a planner should consider. As the number of actions defines a branch factor in the planner, keeping this number to a relevant minimum is very important.

As such, the choice of representation determines much of the performance of the planner and therefore the robot. It is not clear yet what value the proposed ATG models will ultimately have, but I hope that our insights will help other researchers make progress towards more tractable and capable planning.

BIBLIOGRAPHY

[1] Aine, Sandip, Swaminathan, Siddharth, Narayanan, Venkatraman, Hwang, Victor, and Likhachev, Maxim. Multi-heuristic A*. *The International Journal of Robotics Research 35*, 1-3 (2016), 224–243.

[2] Aloimonos, Y., Weiss, I., and Bandyopadhyay, A. Active vision. *Int. Journal of Computer Vision 1*, 4 (1988), 333–356.

[3] Araya, Mauricio, Buffet, Olivier, Thomas, Vincent, and Charpillet, Françcois. A POMDP extension with belief-dependent rewards. In *Proc. of Advances in Neural Information Processing Systems (NIPS)* (2010).

[4] Asfour, T, Regenstein, K, Azad, P, Schröder, J, and Dillmann, R. ARMAR-III: A humanoid platform for perception-action integration. In *Proc., International Workshop on Human-Centered Robotic Systems (HCRS), Munich* (2006), Citeseer, pp. 51–56.

[5] Asfour, Tamim, Schill, Julian, Peters, Heiner, Klas, Cornelius, Bücker, Jens, Sander, Christian, Schulz, Stefan, Kargov, Artem, Werner, Tino, and Bartenbach, Volker. ARMAR-4: A 63 DOF torque controlled humanoid robot. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (2013), IEEE, pp. 390–396.

[6] ASUS. ASUS Xtion PRO LIVE, 2017.

[7] Bajcsy, R. Active perception. *IEEE Trans. on Robotics and Automation 76*, 8 (August 1988), 996–1005.

[8] Ballard, D. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition 13*, 2 (1981), 111–122.

[9] Barry, Jennifer, Hsiao, Kaijen, Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Manipulation with multiple action types. In *Experimental Robotics* (2013), Springer, pp. 531–545.

[10] Barry, Jennifer, Kaelbling, Leslie Pack, and Lozano-Perez, Tomas. A hierarchical approach to manipulation with diverse actions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (2013), IEEE, pp. 1799–1806.

[11] Barry, Jennifer Lynn. *Manipulation with diverse actions.* PhD thesis, Massachusetts Institute of Technology, 2013.

[12] Bellman, Richard. A markovian decision process. Tech. rep., DTIC Document, 1957.

[13] Bernstein, Nicolai Aleksandrovitch. On dexterity and its development. In *Dexterity and its development*, Mark L Latash and Michael Turvey, Eds. Lawrence Erlbaum Associates, Mahwah, NJ, 1996, pp. 1–237.

[14] Bicchi, Antonio. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *Robotics and Automation, IEEE Transactions on 16*, 6 (2000), 652–662.

[15] Bingham, Christopher. An antipodally symmetric distribution on the sphere. *The Annals of Statistics* (1974), 1201–1225.

[16] Bohren, Jonathan, Rusu, Radu Bogdan, Jones, E Gil, Marder-Eppstein, Eitan, Pantofaru, Caroline, Wise, Melonee, Mösenlechner, Lorenz, Meeussen, Wim, and Holzer, Stefan. Towards autonomous robotic butlers: Lessons learned with the PR2. In *2011 IEEE International Conference on Robotics and Automation (ICRA),* (2011), IEEE, pp. 5568–5575.

[17] Botea, Adi, Müller, Martin, and Schaeffer, Jonathan. Near optimal hierarchical path-finding. *Journal of game development 1*, 1 (2004), 7–28.

[18] Brooks, Rodney. Visual map making for a mobile robot. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on* (1985), vol. 2, IEEE, pp. 824–829.

[19] Browatzki, Björn, Tikhanoff, Vadim, Metta, Giorgio, Bülthoff, Heinrich H, and Wallraven, Christian. Active object recognition on a humanoid robot. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)* (2012).

[20] Bruce, James, and Veloso, Manuela. Real-time randomized path planning for robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.* (2002), vol. 3, IEEE, pp. 2383–2388.

[21] Capi, Genci. Robot task switching in complex environments. In *Proc. of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)* (2007).

[22] Cassandra, Anthony R. Optimal policies for partially observable markov decision processes. Tech. rep., Report CS-94-14, Brown Univ, 1994.

[23] Cassandra, Anthony R, Kaelbling, Leslie Pack, and Kurien, James A. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on* (1996), vol. 2, IEEE, pp. 963–972.

[24] Castanon, David A. Approximate dynamic programming for sensor management. In *Proc. of IEEE Conf. on Decision and Control (CDC)* (1997).

[25] Choset, Howie, Lynch, Kevin M, Hutchinson, Seth, Kantor, G, Burgard, Wolfram, Kavraki, Lydia E, and Thrun, Sebastian. *Principles of robot motion: theory, algorithms, and implementation.* Bradford Books, 2005.

[26] Connolly, Christopher I, Burns, JB, and Weiss, R. Path planning using Laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on* (1990), IEEE, pp. 2102–2106.

[27] Connolly, Christopher I, and Grupen, Roderic A. The applications of harmonic functions to robotics. *Journal of Robotic Systems 10*, 7 (1993), 931–946.

[28] Cummings, Jonathan. uBot-7: The design of a compliant dexterous mobile manipulator. Master's thesis, University of Massachusetts Amherst, 2014.

[29] Cummings, Jonathan P, Ruiken, Dirk, Wilkinson, Eric L, Lanighan, Michael W, Grupen, Roderic A, and Sup, Frank C. A compact, modular series elastic actuator. *Journal of Mechanisms and Robotics 8*, 4 (2016), 041016.

[30] Darrell, Trevor, and Pentland, Alex. Active gesture recognition using partially observable Markov decision processes. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on* (1996), vol. 3, IEEE, pp. 984–988.

[31] Deegan, Patrick. *Whole-Body Strategies for Mobility and Manipulation.* PhD thesis, University of Massachusetts Amherst, 2010.

[32] Deegan, Patrick, Grupen, Roderic, Hanson, Allen, Horrell, Emily, Ou, Shichao, Riseman, Edward, Sen, Shiraj, Thibodeau, Bryan, Williams, Adam, and Xie, Dan. Mobile manipulators for assisted living in residential settings. *Autonomous Robots 24*, 2 (2008), 179–192.

[33] Deegan, Patrick, Thibodeau, Bryan J, and Grupen, Roderic. Designing a self-stabilizing robot for dynamic mobile manipulation. Tech. rep., DTIC Document, 2006.

[34] Degallier, S., Righetti, L., Natale, L., Nori, F., Metta, G., and Ijspeert, A. A modular bio-inspired architecture for movement generation for the infant-like robot icub. In *2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics, 2008. BioRob 2008.* (2008), pp. 795–800.

[35] Denzler, J., Zobel, M., and Niemann, H. Information theoretic focal length selection for real-time active 3D object tracking. In *Proc. of the Int. Conf. on Computer Vision* (2003), IEEE, pp. 400–407.

[36] Diftler, Myron A, Mehling, JS, Abdallah, Muhammad E, Radford, Nicolaus A, Bridgwater, Lyndon B, Sanders, Adam M, Askew, Roger Scott, Linn, D Marty, Yamokoski, John D, Permenter, FA, et al. Robonaut 2-the first humanoid robot in space. In *2011 IEEE International Conference on Robotics and Automation (ICRA)* (2011), IEEE, pp. 2178–2183.

[37] Dijkstra, Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[38] Dornhege, Christian, Eyerich, Patrick, Keller, Thomas, Trüg, Sebastian, Brenner, Michael, and Nebel, Bernhard. Semantic attachments for domain-independent planning systems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)* (September 2009), AAAI Press, pp. 114–121.

[39] Durrant-Whyte, Hugh, Rye, David, and Nebot, Eduardo. Localization of autonomous guided vehicles. In *Robotics Research*. Springer, 1996, pp. 613–625.

[40] Eidenberger, Robert, and Scharinger, Josef. Active perception and scene modeling by planning with probabilistic 6D object poses. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2010).

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

[42] Gibson, E.J., and Spelke, E.S. *The Development of Perception*, 4 ed. Wiley, 1983.

[43] Glover, Jared, Bradski, Gary, and Rusu, Radu Bogdan. Monte carlo pose estimation with quaternion kernels and the bingham distribution. In *Robotics: science and systems* (2012), vol. 7, p. 97.

[44] Grabner, Michael, Grabner, Helmut, and Bischof, Horst. Fast visual object identification and categorization. In *Proc. of NIPS Workshop in Interclass Transfer* (2005).

[45] Hadfield-Menell, Dylan. *Execution cost optimization for hierarchical planning in the now.* PhD thesis, Massachusetts Institute of Technology, 2013.

[46] Hadfield-Menell, Dylan, Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Optimization in the now: Dynamic peephole optimization for hierarchical planning. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (2013), IEEE, pp. 4560–4567.

[47] Hart, Peter E, Nilsson, Nils J, and Raphael, Bertram. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107.

[48] Hauser, Kris, Bretl, Timothy, Harada, Kensuke, and Latombe, Jean-Claude. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic Foundation of Robotics VII*. Springer, 2008, pp. 507–522.

[49] Hauser, Kris, and Latombe, Jean-Claude. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research 29*, 7 (2010), 897–915.

[50] Hogan, N. Impedance control - An approach to manipulation. I - Theory. II - Implementation. III - Applications. *ASME Transactions Journal of Dynamic Systems and Measurement Control B 107* (Mar. 1985), 1–24.

[51] Hogman, V., Bjorkman, M., Maki, A., and Kragic, D. A sensorimotor learning framework for object categorization. *Trans. on Autonomous Mental Development* (2015).

[52] Hornung, A., Phillips, M., Jones, E.G., Bennewitz, M., Likhachev, M., and Chitta, S. Navigation in three-dimensional cluttered environments for mobile manipulation. In *IEEE International Conference on Robotics and Automation (ICRA), 2012* (2012), pp. 423–429.

[53] Hsiao, Kaijen, Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Task-driven tactile exploration. In *Proc. of Robotics: Science and Systems (RSS)* (2010), Robotics: Science and Systems Conference.

[54] Ihrke, Chris A, Mehling, Joshua S, Parsons, Adam H, Griffith, Bryan Kristian, Radford, Nicolaus A, Permenter, Frank Noble, Davis, Donald R, Ambrose, Robert O, Junkin, Lucien Q, et al. Rotary series elastic actuator, Oct. 23 2012. US Patent 8,291,788.

[55] Inc., Merriam-Webster. *Merriam-Webster's collegiate dictionary.* Merriam-Webster, 2004.

[56] Jung, Hee-Tae, Baird, Jennifer, Choe, Yu-Kyong, and Grupen, Roderic A. Upper extremity physical therapy for stroke patients using a general purpose robot. In *RO-MAN, 2011 IEEE* (2011), IEEE, pp. 270–275.

[57] Jung, Hee-Tae, Takahashi, Takeshi, Choe, Yu-Kyong, Baird, Jennifer, Foster, Tammie, and Grupen, Roderic A. Towards extended virtual presence of the therapist in stroke rehabilitation. In *2013 IEEE International Conference on Rehabilitation Robotics (ICORR)* (2013), IEEE, pp. 1–6.

[58] Jung, Hee-Tae, Takahashi, Takeshi, and Grupen, Rod. Human-robot emergency response-experimental platform and preliminary dataset. Tech. rep., DTIC Document, 2014.

[59] Kaelbling, Leslie Pack, Littman, Michael L, and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*, 1 (1998), 99–134.

[60] Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Hierarchical task and motion planning in the now. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 1470–1477.

[61] Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Pre-image backchaining in belief space for mobile manipulation. In *International Symposium on Robotics Research (ISRR)* (2011).

[62] Kato, H., and Billinghurst, M. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. of IEEE/ACM Int. Workshop on Augmented Reality (IWAR)* (1999).

[63] Kavraki, Lydia E, and Latombe, Jean-Claude. Probabilistic roadmaps for robot path planning. *Pratical motion planning in robotics: current aproaches and future challenges* (1998), 33–53.

[64] Kavraki, Lydia E, Svestka, Petr, Latombe, J-C, and Overmars, Mark H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation 12*, 4 (1996), 566–580.

[65] Khatib, Oussama. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research 5*, 1 (1986), 90–98.

[66] Knepper, Ross A, and Kelly, Alonzo. High performance state lattice planning using heuristic look-up tables. In *IROS* (2006), pp. 3375–3380.

[67] Koenderink, Jan J, and Doorn, AJ van. The internal representation of solid shape with respect to vision. *Biological cybernetics 32*, 4 (1979), 211–216.

[68] Koenig, Sven, and Likhachev, Maxim. D* lite. *Proceedings of the national conference on artificial intelligence* (2002), 476–483.

[69] Köhler, Wolfgang. *Intelligenzprüfungen an Menschenaffen.* J. Springer, 1921.

[70] Kolhe, Pushkar, Dantam, Neil, and Stilman, Mike. Dynamic pushing strategies for dynamically stable mobile manipulators. In *2010 IEEE International Conference on Robotics and Automation (ICRA),* (2010), IEEE, pp. 3745–3750.

[71] Konidaris, George, Kuindersma, Scott R., Grupen, Roderic A., and Barto, Andrew G. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research 31*, 3 (2012), 360–375.

[72] Kortenkamp, David, and Weymouth, Terry. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *AAAI* (1994), vol. 94, pp. 979–984.

[73] Kosaka, Akio, and Kak, Avinash C. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP: Image understanding 56*, 3 (1992), 271–329.

[74] Ku, L., Learned-Miller, E., and Grupen, R. Modeling objects as aspect transition graphs to support manipulation. In *Proc. of Int. Symp. on Robotics Research (ISRR)* (2015).

[75] Ku, L., Sen, S., Learned-Miller, E., and Grupen, R. Action-based models for belief-space planning. In *RSS Workshop on Information-Based Grasp and Manipulation Planning* (2014).

[76] Ku, L., Sen, S., Learned-Miller, E., and Grupen, R. Aspect Transition Graph: An affordance-based model. In *ECCV Workshop on Affordances: Visual Perception of Affordances and Functional Visual Primitives for Scene Analysis* (2014).

[77] Kuffner, James, Nishiwaki, Koichi, Kagami, Satoshi, Inaba, Masayuki, and Inoue, Hirochika. Motion planning for humanoid robots. In *Robotics Research*. Springer, 2005, pp. 365–374.

[78] Kuffner, James J, and LaValle, Steven M. RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (2000), vol. 2, IEEE, pp. 995–1001.

[79] Kuindersma, Scott R, Grupen, Roderic A, and Barto, Andrew G. Variable risk control via stochastic optimization. *The International Journal of Robotics Research 32*, 7 (2013), 806–825.

[80] Kuindersma, S.R., Hannigan, E., Ruiken, D., and Grupen, R.A. Dexterous mobility with the ubot-5 mobile manipulator. In *International Conference on Advanced Robotics, 2009. ICAR 2009.* (2009), pp. 1–7.

[81] Kuipers, Benjamin, and Byun, Yung-Tai. A robust, qualitative method for robot spatial learning. In *AAAI* (1988), vol. 88, pp. 774–779.

[82] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86.

[83] Kurniawati, Hanna, Hsu, David, and Lee, Wee Sun. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems* (2008), vol. 2008, Zurich, Switzerland.

[84] Lai, Kevin, Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. A scalable tree-based approach for joint object and pose recognition. In *Proc. of AAAI Conf. on Artificial Intelligence* (2011).

[85] Lanighan, Michael W., Takahashi, Takeshi, and Grupen, Roderic A. Planning manual tasks in a hierarchical belief space. In *(in submission) Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2017).

[86] Lauwers, TB, Kantor, George A, and Hollis, RL. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (2006), IEEE, pp. 2884–2889.

[87] Lavalle, Steven M. Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Computer Science Dept., Iowa State University, 1998.

[88] LaValle, Steven M, and Kuffner Jr, James J. Rapidly-exploring random trees: Progress and prospects. *Proceedings Workshop on the Algorithmic Foundations of Robotics* (2000).

[89] Li, Zexiang, Canny, JF, and Sastry, Shankar S. On motion planning for dexterous manipulation. I. the problem formulation. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on* (1989), IEEE, pp. 775–780.

[90] Likhachev, Maxim, and Ferguson, Dave. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research 28*, 8 (2009), 933–945.

[91] Likhachev, Maxim, Ferguson, Dave, Gordon, Geoff, Stentz, Anthony, and Thrun, Sebastian. Anytime dynamic A*: An anytime, replanning algorithm. *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)* (2005), 262–271.

[92] Likhachev, Maxim, Gordon, Geoff, and Thrun, Sebastian. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems (NIPS) 16* (2003).

[93] Loeb, Gerald E, and Fishel, Jeremy A. Bayesian action & perception: representing the world in the brain. *Frontiers in neuroscience 8* (2013), 341–341.

[94] Lu, Zhenli, Lallee, S., Tikhanoff, V., and Dominey, P.F. Bent leg walking gait design for humanoid robotic child-iCub based on key state switching control. In *Robotics and Applications (ISRA), 2012 IEEE Symposium on* (2012), pp. 992–998.

[95] Ma, Raymond R, and Dollar, Aaron M. On dexterity and dexterous manipulation. In *15th International Conference on Advanced Robotics (ICAR), 2011* (2011), IEEE, pp. 1–7.

[96] Ma, Raymond R, Odhner, Lael U, and Dollar, Aaron M. A modular, open-source 3D printed underactuated hand. In *2013 IEEE International Conference on Robotics and Automation (ICRA)* (2013), IEEE, pp. 2737–2743.

[97] Marder-Eppstein, Eitan, Berger, Eric, Foote, Tully, Gerkey, Brian, and Konolige, Kurt. The office marathon: Robust navigation in an indoor office environment. In *IEEE International Conference on Robotics and Automation (ICRA), 2010* (2010), IEEE, pp. 300–307.

[98] Nakamura, Y. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[99] Nau, Dana S, Au, Tsz-Chiu, Ilghami, Okhtay, Kuter, Ugur, Murdock, J William, Wu, Dan, and Yaman, Fusun. Shop2: An HTN planning system. *J. Artif. Intell. Res.(JAIR) 20* (2003), 379–404.

[100] Nvidia. Nvidia Jetson TX1 the embedded platform for autonomous everything, 2017.

[101] Okamura, Allison M, Smaby, Niels, and Cutkosky, Mark R. An overview of dexterous manipulation. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (2000), vol. 1, IEEE, pp. 255–262.

[102] Ott, Ch, Eiberger, Oliver, Friedl, Werner, Bauml, B, Hillenbrand, Ulrich, Borst, Ch, Albu-Schaffer, Alin, Brunner, Bernhard, Hirschmuller, H, Kielhofer, S, et al. A humanoid two-arm system for dexterous manipulation. In *2006 6th IEEE-RAS International Conference on Humanoid Robots* (2006), IEEE, pp. 276–283.

[103] Pineau, Joelle, Gordon, Geoff, Thrun, Sebastian, et al. Point-based value iteration: An anytime algorithm for POMDPs.

[104] Pineau, Joelle, Gordon, Geoffrey, and Thrun, Sebastian. Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research 27* (2006), 335–380.

[105] Pivtoraiko, Mihail, and Kelly, Alonzo. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005). 2005* (2005), IEEE, pp. 3231–3237.

[106] Platt Jr, Robert J. *Learning and generalizing control-based grasping and manipulation skills.* PhD thesis, Citeseer, 2006.

[107] Pratkanis, Anthony, Leeper, Adam Eric, and Salisbury, Kenneth. Replacing the office intern: An autonomous coffee run with a mobile manipulator. In *IEEE International Conference on Robotics and Automation, 2013. Proceedings. 2013* (2013).

[108] Pratt, Gill A, and Williamson, Matthew M. Series elastic actuators. In *Proceedings. 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots'* (1995), vol. 1, IEEE, pp. 399–406.

[109] Quigley, Morgan, Conley, Ken, Gerkey, Brian, Faust, Josh, Foote, Tully, Leibs, Jeremy, Wheeler, Rob, and Ng, Andrew Y. ROS: an open-source robot operating system. In *ICRA workshop on open source software* (2009), vol. 3.

[110] Robotics, Rethink. Baxter Collaborative Robots for Industrial Automation, 2017.

[111] Robotics, Vecna. The BEAR battlefield extraction-assist robot, 2010.

[112] Ross, Stéphane, Pineau, Joelle, Paquet, Sébastien, and Chaib-Draa, Brahim. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research 32* (2008), 663–704.

[113] Roy, Nicholas, Burgard, Wolfram, Fox, Dieter, and Thrun, Sebastian. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (1999), vol. 1, IEEE, pp. 35–40.

[114] Ruiken, Dirk, Cummings, Jonathan P., Savaria, Uday R., IV, Frank Sup, and Grupen, Roderic A. uBot-7: A dynamically balancing mobile manipulator with series elastic actuators. In *(in submission) Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2017).

[115] Ruiken, Dirk, Lanighan, Michael W, and Grupen, Roderic A. Postural modes and control for dexterous mobile manipulation: the UMass uBot concept. In *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2013* (2013), IEEE, pp. 721–726.

[116] Ruiken, Dirk, Lanighan, Michael W, and Grupen, Roderic A. Path planning for dexterous mobility. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)* (2014).

[117] Ruiken, Dirk, Liu, Tiffany Q., Takahashi, Takeshi, and Grupen, Roderic. Reconfigurable tasks in belief-space planning. In *16th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (2016), IEEE, pp. 1257–1263.

[118] Ruiken, Dirk, Wong, Jay Ming, Liu, Tiffany Q., Hebert, Mitchell, Takahashi, Takeshi, Lanighan, Michael W., and Grupen, Roderic A. Affordance-based active belief: Recognition using visual and manual actions. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2016), pp. 5312–5317.

[119] Ruml, Wheeler, and Do, Minh Binh. Best-first utility-guided search. In *IJCAI* (2007), pp. 2378–2384.

[120] Russell, Stuart, Norvig, Peter, and Intelligence, Artificial. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25* (1995), 27.

[121] Sakagami, Yoshiaki, Watanabe, Ryujin, Aoyama, Chiaki, Matsunaga, Shinichi, Higaki, Nobuo, and Fujimura, Kikuo. The intelligent ASIMO: System overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.* (2002), vol. 3, IEEE, pp. 2478–2483.

[122] Sen, Shiraj. *Bridging the Gap between Autonomous Skill Learning and Task Specific Planning.* PhD thesis, University of Massachusetts Amherst, September 2012.

[123] Sen, Shiraj, and Grupen, Rod. Integrating task level planning with stochastic control. Tech. Rep. UM-CS-2014-005, University of Massachusetts Amherst, 2014.

[124] Sen, Shiraj, and Grupen, Roderic A. Manipulation planning using model-based belief dynamics. In *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (2013).

[125] Sen, Shiraj, Sherrick, Grant, Ruiken, Dirk, and Grupen, Rod. Choosing informative actions for manipulation tasks. In *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (October 2011), IEEE, pp. 721–726.

[126] Sen, Shiraj, Sherrick, Grant, Ruiken, Dirk, and Grupen, Roderic A. Hierarchical skills and skill-based representation. In *Workshop on Lifelong Learning from Sensorimotor Experience, AAAI-11* (August 2011).

[127] Simmons, Reid, and Koenig, Sven. Probabilistic robot navigation in partially observable environments. In *IJCAI* (1995), vol. 95, pp. 1080–1087.

[128] Smith, Trey, and Simmons, Reid. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence* (2004), AUAI Press, pp. 520–527.

[129] Smith, Trey, and Simmons, Reid. Point-based POMDP algorithms: Improved analysis and implementation. *arXiv preprint arXiv:1207.1412* (2012).

[130] Somani, Adhiraj, Ye, Nan, Hsu, David, and Lee, Wee Sun. DESPOT: Online POMDP planning with regularization. In *Proc. of Advances in Neural Information Processing Systems (NIPS)* (2013).

[131] Sondik, Edward Jay. *The optimal control of partially observable Markov processes.* PhD thesis, Stanford University, 1971.

[132] Spaan, Matthijs TJ, and Vlassis, Nikos. Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research 24* (2005), 195–220.

[133] Sridharan, Mohan, Wyatt, Jeremy, and Dearden, Richard. Planning to see: A hierarchical approach to planning visual actions on a robot using POMDPs. *Artificial Intelligence 174*, 11 (2010), 704–725.

[134] Stentz, Anthony, and Mellon, Is Carnegle. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation 10* (1993), 89–100.

[135] Stilman, Mike, and Kuffner, James. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal on Humanoid Robotics 2*, 4 (December 2005), 479–504.

[136] Stilman, Mike, Olson, Jon, and Gloss, William. Golem Krang: Dynamically stable humanoid robot for mobile manipulation. In *2010 IEEE International Conference on Robotics and Automation (ICRA)* (May 2010), IEEE, pp. 3304–3309.

[137] Stilman, Mike, Schamburek, J-U, Kuffner, James, and Asfour, Tamim. Manipulation planning among movable obstacles. In *Robotics and Automation, 2007 IEEE International Conference on* (2007), IEEE, pp. 3327–3332.

[138] Sturges, RH. A quantification of machine dexterity applied to an assembly task. *The International Journal of Robotics Research 9*, 3 (1990), 49–62.

[139] Takahashi, Takeshi, Lanighan, Michael W., and Grupen, Roderic A. Hybrid task planning grounded in belief: Constructing physical copies of simple structures. In *International Conference on Automated Planning and Scheduling (ICAPS)* (2017).

[140] Thibodeau, Bryan J, Deegan, Patrick, and Grupen, Roderic. Static analysis of contact forces with a mobile manipulator. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (2006), IEEE, pp. 4007–4012.

[141] Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots 5*, 3-4 (1998), 253–271.

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

[143] Tsagarakis, Nikos G, Morfey, Stephen, Cerda, Gustavo Medrano, Zhibin, Li, and Caldwell, Darwin G. Compliant humanoid COMAN: Optimal joint stiffness tuning for modal frequency control. In *2013 IEEE International Conference on Robotics and Automation (ICRA)* (2013), IEEE, pp. 673–678.

[144] Van Den Berg, Jur, Stilman, Mike, Kuffner, James, Lin, Ming, and Manocha, Dinesh. Path planning among movable obstacles: a probabilistically complete approach. In *Algorithmic Foundation of Robotics VIII.* Springer, 2009, pp. 599–614.

[145] Vukobratović, Miomir, and Stepanenko, J. On the stability of anthropomorphic systems. *Mathematical Biosciences 15*, 1 (1972), 1–37.

[146] Wawerla, Jens, and Vaughan, Richard T. Robot task switching under diminishing returns. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2009).

[147] Wilcox, Brian H., Litwin, Todd, Biesiadecki, Jeff, Matthews, Jaret, Heverly, Matt, Morrison, Jack, Townsend, Julie, Ahmad, Norman, Sirota, Allen, and

Cooper, Brian. Athlete: A cargo handling and manipulation robot for the moon. *Journal of Field Robotics 24*, 5 (2007), 421–434.

[148] Wilt, Christopher Makoto, and Ruml, Wheeler. Cost-based heuristic search is sensitive to the ratio of operator costs. In *Fourth Annual Symposium on Combinatorial Search* (2011).

[149] Wilt, Christopher Makoto, and Ruml, Wheeler. Speedy versus greedy search. In *Seventh Annual Symposium on Combinatorial Search* (2014).

[150] Wimbock, Thomas, Ott, Christian, and Hirzinger, Gerd. Impedance behaviors for two-handed manipulation: Design and experiments. In *2007 IEEE International Conference on Robotics and Automation (ICRA)* (2007), IEEE, pp. 4182–4189.

[151] Wolfe, Jason, Marthi, Bhaskara, and Russell, Stuart J. Combined task and motion planning for mobile manipulation. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2010)* (2010), pp. 254–258.

[152] Wong, J. M., and Grupen, R. Intrinsically motivated multimodal structure learning. In *Proc. of Int. Conf. on Developmental Learning and Epigenetic Robotics (ICDL-EPIROB)* (2016).

[153] Wörgötter, Florentin, Agostini, Alejandro, Krüger, Norbert, Shylo, Natalya, and Porr, Bernd. Cognitive agents – a procedural perspective relying on the predictability of object-action-complexes (OACs). *Robotics and Autonomous Systems 57*, 4 (2009), 420–432.

[154] Wray, Kyle Hollins, Ruiken, Dirk, Grupen, Roderic A, and Zilberstein, Shlomo. Log-space harmonic function path planning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), IEEE, pp. 1511–1516.

[155] Zucker, Matthew, Kuffner, James, and Branicky, Michael. Multipartite RRTs for rapid replanning in dynamic environments. In *Robotics and Automation, 2007 IEEE International Conference on* (2007), IEEE, pp. 1603–1609.