Hierarchical Belief Spaces for Autonomous Mobile Manipulation

Michael W. Lanighan

University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation
Lanighan, Michael W., "Hierarchical Belief Spaces for Autonomous Mobile Manipulation' (2019). Doctoral Dissertations. 1547.
https://doi.org/10.7275/13739497 https://scholarworks.umass.edu/dissertations_2/1547

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
HIERARCHICAL BELIEF SPACES FOR AUTONOMOUS MOBILE MANIPULATION

A Dissertation Presented
by
MICHAEL WILLIAM LANIGHAN

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 2019

College of Information and Computer Sciences
HIERARCHICAL BELIEF SPACES FOR AUTONOMOUS MOBILE MANIPULATION

A Dissertation Presented

by

MICHAEL WILLIAM LANIGHAN

Approved as to style and content by:

_________________________________
Roderic A. Grupen, Chair

_________________________________
Shlomo Zilberstein, Member

_________________________________
Joydeep Biswas, Member

_________________________________
Frank C. Sup IV, Member

_________________________________
James Allan, Chair
College of Information and Computer Sciences
“Robots don’t have to be very intelligent to be intelligent enough. If a robot can follow simple orders and do the housework, ... we would be perfectly satisfied.”

- Isaac Asimov, Robot Visions
There are several people who I would like to acknowledge for their help and assistance throughout my graduate career. This dissertation would not have been possible without the support and guidance of my advisor, Rod Grupen. Being a member of a “full-stack” robotics lab has given me a holistic perspective on robotics that I do not think I could have attained elsewhere. My time at the Laboratory for Perceptual Robotics has been very rewarding; thanks, Rod! I would also like to thank my committee: Frank Sup, Joydeep Biswas, and Shlomo Zilberstein for providing insightful feedback and suggestions which have helped shape this document. Thanks also to Barb Sutherland, Laurie Downey, Eileen Hamel, and Leeanne Leclerc for helping me navigate all of the paperwork and deadlines during my time at UMass.

Many thanks to the talented members (past and present) of the Laboratory for Perceptual Robotics with whom I have had the pleasure of working: Shiraj Sen, Dirk Ruiken, Hee-Tae Jung, Takeshi Takahashi, Tiffany Liu, Jonathan Cummings, Mitchell Hebert, Jay Wong, Uday Savaria, Scott Jordan, Hia Ghosh, Kushal Sahare, Khoshrav Doctor, Geoff Ament, and Debasmita Ghose. Special thanks are due to Takeshi Takahashi for his assistance in testing many of the ideas in this document and to Khoshrav Doctor for help with collecting the last data set in this dissertation.

Above all, I am grateful for the love and support of my family. Thanks to my parents, grandparents, brothers, sister, aunts, uncles and in-laws for encouraging me to explore my interests. A special thanks to my wife, Emily, for her love and support throughout my PhD. Thank you all for encouraging and believing in me—this is for you.
ABSTRACT

HIERARCHICAL BELIEF SPACES FOR AUTONOMOUS MOBILE MANIPULATION

MAY 2019

MICHAEL WILLIAM LANIGHAN
B.S., CANISIUS COLLEGE
M.S., UNIVERSITY OF MASSACHUSETTS AMHERST
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic A. Grupen

Autonomy in robot systems is a valuable attribute that remains an elusive goal. Noisy sensors, stochastic actions, and variation in unstructured environments all lead to unavoidable errors that can be inconsequential or catastrophic depending on the circumstances. Developing techniques capable of mitigating uncertainty at runtime has, therefore, been a significant and challenging focus of the robotics community.

The primary contribution of this dissertation is the introduction of a new hierarchical belief space planning architecture to manage uncertainty and solve tasks using a uniform framework. Such an approach provides a means of creating autonomous systems that focus on salient subsets of state information, mitigate risk, and require less frequent intervention. Results indicate that it is possible to implement near optimal solutions to interesting problems in a uniform, hierarchical framework of belief space
planners by taking actions that condense belief towards goal distributions. Example hierarchies are presented to address simple assembly problems and to enable robust long-term autonomous mobile manipulation in deployments lasting on the order of hours during which the robot executes hundreds of actions.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vi</td>
<td></td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
<td></td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xii</td>
<td></td>
</tr>
</tbody>
</table>

## CHAPTER

1. INTRODUCTION ................................................. 1
   1.1 Contributions ........................................... 2
   1.2 Outline .................................................. 3

2. BACKGROUND AND RELATED WORK ...................... 6
   2.1 Belief Space Primer ................................. 6
       2.1.1 Maintaining and Updating Belief Distributions 6
       2.1.2 Measuring Uncertainty in Belief Distributions 8
   2.2 Task and Motion Planning ............................. 11
   2.3 Error Detection and Recovery ....................... 13
   2.4 Long-term Autonomy .................................. 14
   2.5 Architectures for Autonomy ........................... 17
   2.6 Belief Space Planning ............................... 18
       2.6.1 Active Perception and Active Belief ............ 19
   2.7 Experimental Apparatus ............................... 23
       2.7.1 Experimental Objects: ARcubes .................. 24
3. HYBRID PLANNING GROUNDED IN BELIEF

3.1 Hybrid Task Planner
3.2 Partitioning the Belief Space
  3.2.1 Finding an Object of the Target Class
  3.2.2 Orienting an object
3.3 Symbolic Planning and Symbol Grounding
3.4 Demonstration Platform and Domain
  3.4.1 Assemblies
3.5 Outcomes
3.6 Discussion

4. HIERARCHICAL BELIEF SPACE PLANNING

4.1 Hierarchical Belief Space Planning
  4.1.1 Example State Abstraction Function
4.2 Confidence-based Subsumption in Belief Space
4.3 A Two-Level Hierarchy for Assembly
  4.3.1 Object Level
  4.3.2 Assembly Level
  4.3.3 Control Authority: ARBITER
4.4 Example Application: Simple Assemblies
4.5 Experimental Results
  4.5.1 Comparison to Baseline Planner
  4.5.2 Pyramid Assembly
  4.5.3 Performance in Face of External Perturbations
4.6 Discussion

5. LONG-TERM AUTONOMY UNDER UNCERTAINTY

5.1 Environmental Level
  5.1.1 Harmonic Function Planning and Information Gain
5.2 Intra-object Level
5.3 Inter-object Level
5.4 ARBITER
5.5  Vanishing Belief ................................................................. 66

5.5.1  Managing Vanished Belief Distributions ......................... 68

5.6  Evaluating Long-Term Autonomy under Uncertainty ............... 68
5.7  Implementation Details ................................................. 70
5.8  Deployment Results ..................................................... 70

5.8.1  Interventions ......................................................... 72
5.8.2  Experimenter Interactions ........................................... 74

5.9  Discussion ................................................................. 78

6.  CONCLUSION ................................................................. 80

6.1  Recommendations for Future Work ..................................... 81

6.1.1  Autonomous Model Building ......................................... 82
6.1.2  Arbitration Mechanisms .............................................. 83
6.1.3  Explainable AI .......................................................... 83
6.1.4  Recovering Vanished Belief Distributions ......................... 84
6.1.5  Temporal Planning in Belief Space ................................. 84
6.1.6  Additional Extensions to the Three-Level Hierarchy ........... 85

BIBLIOGRAPHY ................................................................. 87
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Computed entropy and variance measures for the distributions of Figure 2.2. Entropy captures the greater amount of uncertainty present in the distribution Belief 1, while variance does not.</td>
</tr>
<tr>
<td>4.1</td>
<td>The mean ($m$) and variance ($m^2$) of final error ($\ell_2$ distance) in tower experiments (shown in Figure 4.4) using the baseline and the hierarchical framework (HABP). Lower error values indicate better performance. As $p &lt; 0.05$, the results show statistically better performance of HABP approach compared to the baseline.</td>
</tr>
<tr>
<td>5.1</td>
<td>The duration, number of actions executed by the robot, items correctly stowed, and the number of external interventions that were required (failures) across 7 deployments.</td>
</tr>
<tr>
<td>5.2</td>
<td>Breakdown of the 154 actions executed over seven deployments by hierarchy level.</td>
</tr>
<tr>
<td>5.3</td>
<td>The number of interactions taken by the experimenter to alter the current environmental state during execution. These interactions moved, withdrew, or added objects in the scene. Despite these attempts to foil the robot, it was able to manage the induced uncertainty with the three-level hierarchy to still address the task. *In deployment 7, three experimenter interactions did not effect the robot at all, as two interactions inserted and withdrew an unobserved object. The third caused a disturbance that was not observed due to an un-related failure that ended the trial prematurely.</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Example of an error in a grasping action (a-b) leads to a cascade of failure from which external intervention is required (c-f)</td>
<td>3</td>
</tr>
<tr>
<td>2.1 A graphical depiction of a belief update. The leftmost distribution, at time $k - 1$ is the <em>prior</em>. The effect of executing action $a$ at time $k - 1$ on the prior is computed using the <em>control update</em>, yielding the middle distribution. The predicted outcomes are corrected through a <em>measurement update</em> using observation $z_k$, resulting in the <em>posterior</em> distribution on the right</td>
<td>8</td>
</tr>
<tr>
<td>2.2 The two distributions clearly exhibit different amounts of uncertainty. However, measures such as variance will fail to capture the difference between the two distributions</td>
<td>10</td>
</tr>
<tr>
<td>2.3 A partial Aspect Transition Graph (ATG) of a toy block. Constellations of visual features present on the block (the numbers on each face) from given sensory configurations define each Aspect node. The model captures the stochastic nature of outcomes of two actions from the Aspect node on the left, probabilistically leading to different outcome states</td>
<td>21</td>
</tr>
<tr>
<td>2.4 uBot-6: a 13 degree of freedom, dynamically balancing whole body mobile manipulator</td>
<td>24</td>
</tr>
<tr>
<td>2.5 Two views of ARcubes illustrating the sparseness of features present from different views. On the left (a), only 2 features are visible, while on the right (b), only 3 features are visible. This sparseness of features leads to a large degree of ambiguity within a model-set of ARcubes as sub-sets of features can be shared across multiple objects</td>
<td>25</td>
</tr>
<tr>
<td>3.1 Example of an ARcube assembly problem generated during execution. The :goal statement is generated when the robot registers the demonstrated assembly. The :init and :object statements are updated after each symbolization</td>
<td>31</td>
</tr>
</tbody>
</table>
3.2 The left figure shows the goal of the copy task. The building blocks that can be used to copy the structure are displayed on the right. Although not visible from this view, hypothesis $h_2$ has both aspects ‘1-x’ (an aspect with ‘1’ in front) and ‘3-0’ (an aspect with ‘3’ in front and ‘0’ on top), while hypotheses $h_0$ and $h_1$ only have aspect ‘1-x’. This introduces a resource constraint as only hypothesis $h_2$ can be used for the top object in the structure.

3.3 On the left is the demonstrated stack, while the right stack is an approximate copy constructed by the robot.

3.4 The figures show the task beliefs for each hypothesis in the scene for finding aspect ‘1-x’ (top figure) and aspect ‘3-0’ (bottom figure). The robot symbolized all of the hypothesis with aspect ‘1-x’ after the first observation at 423 seconds, and symbolized hypothesis $h_2$ with aspect ‘3-0’ at 759 seconds.

4.1 Graphical representation of data flow in a hierarchy of $n$ levels. Belief from the lower levels is used as inputs in higher level planners. Each level of the hierarchy $j$ selects actions $a_j^*$ to execute. An arbitration module named the Arbiter selects an action $a^*$ to execute from the set of prescribed actions.

4.2 Graphical representation of an arbitrary level of the hierarchy. Belief from the lower level and the previous time step state of this level are fused into a current state estimate (left side). This state estimate is used to plan forward into the future using $T_i$ and $A_i$ (right side).

4.3 Graphical representation of the confidence-based subsumption Arbiter for a hierarchy of arbitrary depth $n$. When enough confidence exists in higher levels they suppress the outputs of lower-level planners.

4.4 Final tower assemblies produced by the two approaches in the first experiment (comparing to a baseline). On the left are the five assemblies using the baseline approach. Due to uncertainty and stochastic actions the assemblies created with this approach often fail. On the right are the outcomes using the HABP framework. By managing uncertainty and risk the planner is able to construct better approximations of the goal assembly.
4.5 Example pyramid assembly using the proposed approach. Each picture is a snapshot of the assembly from the robot’s perspective during the task. In between each image the robot places new objects into the assembly or replaces existing objects if it predicts it is capable of achieving a better outcome. .......................... 54

4.6 Outcomes of intentional perturbation by an experimenter during a pyramid assembly. The robot begins to construct the assembly (a-b). The robot is no longer confident it has placed the object in the correct orientation after the unexpected action outcome (c-e). The robot re-configures the planner to reduce this uncertainty (f-h). After the uncertainty has been addressed, the robot replaces the object in the assembly in the correct orientation (i-k) then moves to complete the rest of the assembly (l) ...................... 56

5.1 Data flow in a three-level HABP architecture. Belief stabilizes as it progresses up the hierarchy. Each planner selects actions $a^*$ to execute that maximize their respective metrics. An arbitration module selects an action from the set of best actions to execute on the robot. .......................................................... 59

5.2 Snapshots showing an example trajectory roll-out of a 2D robot (in red) exploring a room using the Frontier-based harmonic function path planner. The trajectory gains information while simultaneously maximally avoiding obstacles (in blue) when unknown areas (in green) are used as goals. The trajectories direct the robot towards the largest unknown areas remaining during search. ................................................................. 63

5.3 Images showing the failures experienced during deployments. Only the failure shown in (b) is attributable to the HABP system. A bug in the implementation prevented messages from being transferred between planning processes. This was rectified in subsequent deployments. ............................................................. 73

5.4 The sole failure after experimenter interaction (b-e) during the first deployment. Due to an implementation error, although the environmental level detected that the object had been removed the intra-object level belief was not correctly updated, causing the robot to grasp empty space (g), which required intervention to correct. ................................................................. 76
5.5 Example behavior after experimenter interaction. After observing the object in (a), the robot executes an action to observe the object from another vantage point (b-e). An experimenter removes the object in (c). By sampling the environmental level’s belief of occupancy (which provided little support for the expected volume), the intra-object level determined the object had been removed (e). Execution was then yielded to the environmental level to find new objects (f).

5.6 Example robust behavior after experimenter interaction due to reasoning in the hierarchy. While manipulating an object (a-d) an experimenter removes the object (c-d). By sampling the environmental level’s belief of occupancy, the intra-object level determined that the object had been removed, as no support for a volume existed in the expected position (e). The intra-object level execution was then yielded to the environmental level, which began executing trajectories to find new objects to interact with (f).
CHAPTER 1
INTRODUCTION

Robots have long held our imagination as ubiquitous helpers, aides, and personal assistants aiding us in our daily lives. Despite advances in manipulation, locomotion, and perception, results from recent robot challenges indicate that robots able to cope reliably with unstructured and, therefore, partially observable environments (akin to Rosie of *The Jetsons*) remains an elusive goal [46, 17]. Unfortunately, such robust systems have only occupied the realm of science fiction. Why is this? Although large strides have been made in robotics—particularly in the development of high-performance, high degree-of-freedom robot systems—when inserted into real-world, unstructured environments they often fail in unexpected and unforeseen ways.

This is due primarily to *uncertainty* present in unstructured environments, sensing, and actuation. Uncertainty can introduce enough risk over different instances of the same task that makes traditional planning approaches impractical. Developing planning and execution frameworks able to overcome uncertainty and manage risk is, therefore, a major goal of robotics.

Noisy sensor readings and stochastic actions are common in robotics. Unstructured environments introduce hidden state and uncertainty that make decisions about control more difficult. In complex tasks that employ tens or hundreds of actions in a sequence, a single fault can disable the robot and/or damage the environment. Because of this, robots deployed in unstructured environments for long-term missions—such as the Mars rovers—often have limited autonomy [25]. This lack of reliable autonomy limits the performance of robot systems deployed in unstructured environ-
ments. In the case of the Mars rovers, when even modest autonomous behavior is introduced into these systems the performance (in terms of the quantity and quality meaningful samples collected) can be significantly increased [24].

In order to embed autonomous robots in unstructured worlds, it seems reasonable to specify a reliability in excess of thousands of control decisions between failures, where a “failure” denotes a situation that requires external reset. To meet these goals, breakthroughs are required concerning the assessment of uncertainty—specifically as it puts a task at risk—and in the formulation of risk averse and error recovery behavior.

The sensitivity of performance to undetected or hidden state is demonstrated by the performance of tele-operated and autonomous robots in the literature [46, 17]. Some of the results of the DARPA Robotics Challenge (DRC) illustrated how state of the art robots can be foiled by uncertainty in the current state and/or the stochasticity of actions. For example, in tasks such as opening a door, crossing the threshold, and turning valves often caused unanticipated catastrophic outcomes. Figure 1.1 shows the JAXON humanoid [45] attempting to turn a valve when errors during the initial grasp (Figures 1.1a-1.1b) lead to a complete failure (Figures 1.1c-1.1f). If the robot was able to detect inappropriate state transitions early in the task, such as during Figure 1.1b-1.1c, it may have been able to recover instead of suffering complete failure. Representations of expected state transitions which support prediction and planning techniques that can choose actions from unanticipated, novel states are open problems.

1.1 Contributions

The goal of this dissertation is to contribute to autonomous decision making in robots that interact with partially observable, unstructured environments. To this end, the contributions are:
Figure 1.1: Example of an error in a grasping action (a-b) leads to a cascade of failure from which external intervention is required (c-f).

1. A hybrid planning framework to accomplish tasks using symbolic planning that leverages robust interaction via belief space planning methods.

2. A recursive definition of model-based belief space planners that provides a basis for hierarchical organization and supports multiple levels of abstraction.

3. Example hierarchies that manage uncertainty in autonomous behavior that operates reliably for an extended period of time (on the order of hours) without external intervention.

1.2 Outline

This dissertation is organized as follows: In Chapter 2 the state of the art in fields relevant to this proposal is summarized. A primer on basic concepts of reasoning in belief space is provided in section 2.1. Techniques commonly used in robotics to
address problems like those we will investigate known as task and motion planning will be surveyed in Section 2.2. Section 2.3 overviews error detection and recovery. A discussion of research in techniques to enable long-term autonomy will be surveyed in Section 2.4. Section 2.5 will provide a brief survey on architectures for autonomy and describes into what paradigm the work in this thesis falls. Section 2.6 introduces techniques for belief space planning, the Aspect Transition Graph representation, and the Active Belief Planner (ABP), which will form the basis of the hierarchical planning architecture in this dissertation. Finally, Section 2.7 introduces the uBot-6 mobile manipulator, the experimental platform used throughout this dissertation and the ARcube domain used in experiments.

Chapter 3 highlights work in hybrid planning that motivates the push for hierarchical belief spaces. This framework was a first attempt to leverage belief space planning to make robots more reliable while completing tasks. To achieve this, a belief space planner (the ABP) actively suppresses the influence of object-level uncertainty so a traditional PDDL based planner can be reliably used on a robot. This is accomplished by grounding environmental percepts into symbols based upon the belief of those percepts drawn from a model-set. Experimental results in a simple assembly domain with the uBot-6 mobile manipulator demonstrate the benefits of this approach over maximum likelihood approaches. Portions of Chapter 3 are found in Takahashi, Lanighan, and Grupen [73].

The hybrid system of Chapter 3 encountered difficulties when real-world dynamics did not match the deterministic dynamics of the symbolic domain. This shortcoming motivated extending the belief space planning framework into a hierarchal system. Chapter 4 outlines the theory behind the Hierarchical Active Belief Planner framework and provides experimental results of a two-level hierarchy implementation used for assembly planning. Statistically significant increases in performance compared to the approach of Chapter 3 are found. Additionally, demonstrations highlighting the
robust behavior resulting from the framework are presented. Portions of Chapter 4 are found in Lanighan et al. [52].

Evaluating the hierarchical approach in settings of long-term autonomy is explored in Chapter 5. A three-level hierarchy which overcomes shortcomings of the two-level assembly hierarchy is proposed. This system is evaluated in a dynamic tidy-up domain, where a robot is tasked to clean an area that is continuously untidied by an experimenter. To do so, the robot actively seeks information to: (1) accumulate knowledge of environmental occupancy; (2) detect and identify objects that satisfy task-specific roles while ignoring distractors; and (3) control uncertainty in the task geometry. The system is evaluated by measuring the number of manual interventions needed during the deployments, which last several hours over which hundreds of actions are executed by the robot. Chapter 6 concludes the dissertation and provides recommendations for future research.
CHAPTER 2
BACKGROUND AND RELATED WORK

This dissertation describes methods for reasoning in belief spaces to enable more robust robotic systems. Critical to understanding the contributions of this dissertation is understanding core concepts of belief space techniques and how they can be applied. This chapter will introduce basic principles to the reader to ease understanding of later chapters. Sections following the primer survey areas related to this dissertation.

2.1 Belief Space Primer

Consider a random variable $X$, which could correspond to the position of a robot along a one-dimensional line. Assume this variable can take values from the interval between values $\alpha$ and $\beta$. The probability that variable $X = x$ where $x \in (\alpha, \beta)$ is captured by a probability density function $f(x)$. This distribution is referred to as a belief distribution as it captures the likelihood or belief that $X = x$ for any $x \in (\alpha, \beta)$.

A belief state $b_t$ describes the probabilistic state (the distribution) of the system at a specific time $t$. In the interval between $(\alpha, \beta)$ an infinite number of possible belief distributions exist. The space of all belief distributions is referred to as belief space and is usually denoted as a set $B$.

2.1.1 Maintaining and Updating Belief Distributions

An initial belief distribution is known as a prior distribution. This can be formed from previous experiences or can be arbitrarily set. The prior belief at time $t$ of a state
$x$ can be defined by the belief state at time $t$, $b_t$. Given an initial belief state, we need to update belief when actions are executed and new observations of the environment are made.

At time $t$, the robot may execute an action $a_t$. A transition model defined $\Pr(x_{t+1}|a_t, x_t)$ can inform the robot of the effects of its action. The effect of taking an action on the prior can be predicted with

$$\overline{b}_{t+1} = \int_x \Pr(x_{t+1}|a_t, x_t) \, b_t \, dx_t$$

(2.1)

where $\overline{b}_{t+1}$ is the predicted distribution due to executing action $a_t$. The prediction step in equation 2.1 is known as a control update. Specifying and obtaining such transition models for generic actions and objects is an open research question, but can be obtained empirically or learned [47, 79] and will be discussed later. In discrete domains the control update is described by the sum

$$\overline{b}_{t+1} = \sum_x \Pr(x_{t+1}|a_t, x_t) \, b_t$$

(2.2)

This new predicted belief state will account for all plausible outcomes of the action taken, therefore, it will disperse the previous belief amongst these outcomes. To recover the true belief state, $\overline{b}_{t+1}$ from equations 2.1 and 2.2 needs to be corrected through additional observations. An observation model $\Pr(z_{t+1}|x_{t+1})$ can be used to determine how likely an observation $z_{t+1}$ supports the new state. The new evidence is incorporated through

$$b_{t+1} = \eta \, \Pr(z_{t+1}|x_{t+1}) \, \overline{b}_{t+1}$$

(2.3)

where $b_{t+1}$ is the posterior belief distribution and $\eta$ is a normalization constant. Equation 2.3 is known as the measurement update and corrects the predicted state with new observations. The posterior belief $b_{t+1}$ can then be used as the prior for
the next time step. This process is known as recursive Bayesian filtering and yields robust state estimation. A graphical depiction of a single belief update is shown in Figure 2.1.

![Belief Update Diagram](image)

Figure 2.1: A graphical depiction of a belief update. The leftmost distribution, at time \( k - 1 \) is the prior. The effect of executing action \( a \) at time \( k - 1 \) on the prior is computed using the control update, yielding the middle distribution. The predicted outcomes are corrected through a measurement update using observation \( z_k \), resulting in the posterior distribution on the right.

The algorithm for performing one update in a discrete domain is outlined in Algorithm 1. This computes the control update for each possible underlying state to form the full belief distribution (Lines 2-3). The measurement update then corrects the predicted distribution with the weight of the new observation \( z_k \) (lines 4-5). The posterior distribution is then returned.

### 2.1.2 Measuring Uncertainty in Belief Distributions

Given belief distributions, it is useful to quantify the amount of uncertainty contained in the distribution. Common measures from statistics such as variance capture uncertainty in unimodal distributions well, however fail to capture uncertainty in multimodal distributions. Consider the distributions shown in Figure 2.2.
Algorithm 1 Bayes Filter: Algorithm for updating the state of a planner using the previous state $b_{k-1}$, action $a_{k-1}$, and new observation $z_k$. $\eta$ is a normalization constant.

1: function Bayes($b_{k-1}, a_{k-1}, z_k$)
2:     for all $s_k$ do
3:         $\overline{b}_k(s_k) = \sum_{s_{k-1}} \Pr(s_k|a_{k-1}, s_{k-1})b(s_{k-1})$
4:     for all $s_k$ do
5:         $b_k(s_k) = \eta \Pr(z_k|s_k)\overline{b}_k(s_k)$
6: return $b_k$

Although the distribution Belief 1 contains much more uncertainty than that of Belief 2, the variance of the two is equivalent (shown in Table 2.1). Belief 1 contains more uncertainty as the true class identity is distributed over a large number of possible values, while Belief 2 is contained within two narrow modes. Other measures are needed to capture this difference.

Information theory provides such useful measures. The foundational measure of information theory is Shannon Entropy, $H(X)$, defined

$$H(X) = - \sum_x \Pr(x) \log_2(\Pr(x)) \quad (2.4)$$

which captures the amount of uncertainty in a probability distribution. Alternatively, Shannon Entropy can be thought of conveying the amount of information present in a distribution. Using a logarithm with base 2 yields units of bits. $H(X) = 0$ indicates that the distribution contains no uncertainty, that is, outcomes are deterministic. Given the distributions in Figure 2.2, Table 2.1 shows that Shannon Entropy captures the greater uncertainty in the distribution Belief 1 compared to Belief 2. We will use the terms Shannon Entropy and entropy interchangeably throughout this dissertation.

Measures can be derived from Shannon Entropy to measure changes in entropy between two distributions. This is commonly referred to as Information Gain. Unfortunately, information gain can refer to one of many measures that captures changes in information. The most basic is related to Mutual Information, and computes the
Figure 2.2: The two distributions clearly exhibit different amounts of uncertainty. However, measures such as variance will fail to capture the difference between the two distributions.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Belief 1</th>
<th>Belief 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance</td>
<td>1.01</td>
<td>1.01</td>
</tr>
<tr>
<td>Entropy</td>
<td>4.08</td>
<td>2.47</td>
</tr>
</tbody>
</table>

Table 2.1: Computed entropy and variance measures for the distributions of Figure 2.2. Entropy captures the greater amount of uncertainty present in the distribution Belief 1, while variance does not.
change in entropy between two time steps conditioned on an action, shown in equation 2.5.

\[ IG = H(X_k) - \mathbb{E}_{x_{k+1}}[H(x_{k+1}|a_k)] \]  

(2.5)

In this dissertation we will use equation 2.5 when referring to Information Gain. IG has the property that when the two distributions have equivalent entropies IG will be zero. In this context, it indicates no further information can be gained through additional interactions. The other measure commonly used to describe changes in entropy is Kullback-Leibler Divergence, \( D_{KL} \). Kullback-Leibler Divergence is defined

\[ D_{KL}(E \parallel G) = \int_{-\infty}^{\infty} e(x) \log \left( \frac{e(x)}{g(x)} \right) dx \]  

(2.6)

where \( e \) and \( g \) represent the densities of \( E \) and \( G \) respectively. Similar to equation 2.5, the divergence is zero when the two distributions are equivalent. This is a nuanced distinction—equation 2.6 will only be zero when the two distributions have the same shape. Therefore, Equation 2.6 can be used to measure how far off from a target distribution a given distribution is. For more background on probabilistic techniques, the reader is directed to Thun et al. [75].

2.2 Task and Motion Planning

Building robot systems capable of successfully completing tasks has long been a goal in robotics, dating back to the Shakey project at SRI [53]. In general, approaches to address this problem are known collectively as task and motion planning approaches. The difficulty of this area arises in that although high-level solutions to a given task might exist (e.g. pick-up the red object from the table), it may not be realizable in the current environmental context (e.g. the object may be inaccessible to manipulation due to other objects in the environment). To overcome this, compli-
mentary reasoning must be performed on the task and on the motions to be executed by the robot.

Several groups are investigating hybrid task planning approaches [78, 26, 71] that divide the task into a symbolic task planning problem and a motion planning problem. The symbolic planner finds high-level solutions to a given problem, while the motion planners provide guarantees regarding path planning and collision avoidance. To transfer knowledge between these independent processes an interface is used to translate symbols into actions and constraints from the motion planner to symbols. If a solution prescribed by the symbolic planner is not feasible for the motion planner to execute (e.g. the object to be grasped is blocked), the motion planner reports a failure and adds a constraint to the symbolic planner, which then re-plans with updated state.

However, these approaches do not address partially observable environments or handle the stochasticity of actions. If an action is possible but fails, these approaches must find another solution in the symbolic domain, then re-plan motion constraints. Additionally, without special handling of possible outcomes it is possible that a failed action may lead to a loss of previously initialized symbols. A notable exception is the work of Kaelbling et al. [41]. They proposed integrating task and motion planning in belief space to address uncertainties that arise in both domains. Using pre-image back-chaining they plan over factored representations of belief, regressing these factors at run-time as required to achieve the task. This representation builds information gathering into problem solving, as the robot gathers supporting evidence for high-level concepts before progressing in a task. The approaches proposed and evaluated in this dissertation are similar to these techniques, but differ in how belief dynamics are generated. Also notable is that Kaelbling et al. use a monolithic planner that decomposes tasks hierarchically. We will use a hierarchy of planners that reason in different belief spaces that are informed by preceding levels. This is described in more
detail in Chapter 4. Hadfield-Menell et al. also use belief space planning to address task and motion planning problems [27]. However, they couple a belief space approach with a classical symbolic planner, similar to the approach described in Chapter 3.

2.3 Error Detection and Recovery

The most sophisticated examples of error recovery in unstructured tasks involve a (small set) of reset conditions that are learned or defined a priori by the user. Detection of these conditions causes a state change to a (small set of) task-level states where appropriate responses to the fault proceed [3, 37]. These systems rely on a mixture of human and automated intervention when errors are detected. Examples of autonomous robot control in the literature often require that autonomous behavior be anticipated by the system designer, leading to problems when system reliability depends on subtle details of the runtime environment. In these cases, models of expected behavior can be used to compare predicted future states to observations and, thus, to identify cases when plans may fail unexpectedly.

Di Lello et al. use Bayesian nonparametric time series models to detect unexpected outcomes of actions during execution [19]. However, they do not propose mechanisms for automatic recovery. Rodriguez et al. build empirical models called “grasp signatures” which are used with PCA and a Bayesian SVM to detect grasps that lead probabilistically to failure [60]. The authors demonstrated an “early abort and retry” recovery technique using Markov chains that reset the system to a fixed state if the predicted grasps are likely to be unsatisfactory. Similarly, Ku et al. demonstrate how Aspect Transition Graphs can be used to predict when runtime observations refute the assumed state of manipulation tasks and avoids the unintended consequences of future actions by aborting executions [49]. Kappler et al. used similar manipulation graphs to manage errors during execution by aborting and re-attempting failed actions [43].
Paolini et al. build probabilistic models of the performance of actions by empirically modelling sensors (grasps) and task requirements offline for three different types of “post-grasp” manipulation (placing) tasks [55]. These models are then combined online to predict the outcome of task execution based on the current state. They note that this framework could be extended to select an optimal action from a set of possible actions based on the current (probabilistic) state or abort runs where the probability of success is low.

Some research groups have also taken a shared autonomy approach to recognize and overcome errors at runtime. These approaches rely on human input to correct and recover from failures at runtime. Sankaran et al. propose an architecture that predicts logic based failures and leverages expert human input to recover from action and perception failures [67]. Tellex et al. propose using inverse semantics to generate meaningful and directed help requests to a human in order to recover from failures at runtime [74].

2.4 Long-term Autonomy

Autonomy comes from the Greek autonomos, meaning “having its own laws” from autos “self” and nomos “laws.” An autonomous system is, therefore, one that makes its own decisions according to its own prerogatives. Research in long-term autonomy (LTA) focuses on developing technologies to extend the viability of autonomous systems in long-term deployments. As such, the field is better described by the phrase long-term viability of autonomous systems.

The majority of robots present in the world today execute reliably in factories over long-term deployments. However, these robots operate under a closed world assumption, i.e. the complete state of the environment and its dynamics are assumed to be completely known. These environments are highly engineered and structured so as to effectively eliminate uncertainty. As a result, these robots can often achieve long-
term executions using static world models and strategies. Failures in these domains are attributable to failures in hardware rather than from errors in decision making or internal representations.

Research in LTA focuses on developing technologies to address problems introduced by open worlds [51]. Open worlds are defined by a lack of structure, partial observability, and dynamic environments. Over deployments, environments can (and likely will) evolve and change, rendering static models, policies, and schedules ineffective. The concepts of closed and open worlds are analogous to the ideas of closed and open systems in thermodynamics. In a closed system, the laws of thermodynamics are guaranteed to hold and the complete state of the thermodynamic system can be described through the laws of the thermodynamics. In an open system matter can be introduced and removed. As such, thermodynamic laws no longer hold in general. Additional mechanisms are required to handle such systems. Similarly, additional mechanisms are needed to address the issues introduced by open worlds.

Developing techniques able to cope with these issues is a hallmark of research in LTA. Open-world domains studied in the context of LTA include space (satellites and rovers) [12], marine (autonomous underwater vehicles and gliders) [38, 54], roads (autonomous cars) [36], and mobile service robots [6, 29]. Although the overall goal—long-term viability of autonomous systems—is the same in each domain, nuances of each domain motivate specific research directions. Specifically, the barriers preventing long-term viability in terms of the estimation and control of uncertainty and risk during execution varies across the domains.

Communication delays in space domains motivates the development of technologies able to dynamically re-schedule tasks when assumed models no longer hold [12]. This approach does not address partial observability, but does help extend the long-term viability of the overall system by continuing operations after a task failure. LTA work in mobile robot domains focus on developing localization and mapping
techniques to effectively update internal representations [6] and model environmental
dynamics [29] to allow for completion of user-specified high-level tasks. LTA research
in autonomous vehicles aims to increase LTA by focusing on situations (such as in-
tersections) that challenge state of the art autonomous vehicles [80].

A major missing component in systems being studied in the context of long-
term autonomy is physical interaction with the environment. Studies in long-term autonomy do not address challenges that arise when the robot intentionally alters
the environment. The best examples of long-term deployments of (semi-)autonomous mobile manipulators and the issues they encounter are the systems deployed at the DARPA Robotics Challenge. These systems often experienced failures that required external interventions within tens of actions due to a low mean-time between decision making failures [46]. The notable exception of Team WPI-CMU (which completed their runs without external resets) can be attributed to implemented safety monitors and a ‘slow-and-steady’ approach [3]. This is a promising step in the right direction. Work is needed to generalize this idea to allow a system to autonomously monitor its internal state and take actions to minimize uncertainty and risk as they put tasks at risk.

In Chapter 5 we will evaluate the hierarchical belief architecture of this dissertation in long-term deployments. As LTA domains vary greatly, so does the concept of what duration denotes a “long-term deployment”—from hours [6] to days [29] to years [12]. In this work, we view “long-term deployments” as autonomous deployments that last on the order of hours during which 100s of actions that involve interactions that intentionally alter the environment are executed. Performance will be evaluated by quantifying the number of tasks completed within a time-frame, the number of actions executed, and the number of interventions required due to decision making failures.
2.5 Architectures for Autonomy

Initial architectures for autonomous robot behavior stemmed from early work on Shakey [53]. These initial strategies came to be known as sense–plan–act architectures. In it, the agent uses sensors to detect the current world state and matches it to internal models. These models are then used to plan actions, which are then executed. This approach allowed for ease in task-level planning, however, it has several shortcomings, including that planning in any interesting real-world settings would take noticeable time, which causes the robot to “hang” while planning at run-time. More detrimental though, is that execution of a prescribed plan did not take sensing into account. This would lead to failures in dynamic or noisy environments.

To overcome these issues, paradigms known as reactive planning emerged [7, 21]. These architectures addressed the shortcomings of sense-plan-act approaches by adapting plans and behaviors at runtime based on sensory feedback. Most well-known of these approaches is the subsumption framework of Brooks [7]. This architecture used multiple levels of simple behaviors—higher levels could inhibit lower levels or add signals to their outputs—to make more complex behaviors. This combination/inhibition was enabled through an arbitration mechanism. Although this approach was lauded for the robust behaviors it generates—that would have been impossible with the sense-plan-act model—specifying higher level tasks with the approach was difficult.

In response, hybrid approaches that combined the strengths of these two areas emerged. They are often referred to as layered architectures as they structured functionality and reasoning across multiple levels. The 3T architecture of Bonasso et al. is an exemplar of this type of approach [57]. 3T decomposed problems across three levels: a set of reactive skills to interact with the environment, a sequencer to manage skill activation to accomplish tasks, and a deliberative planner to specify tasks. Using this structure plans can be adapted at runtime to ensure successful completion.
However, as skills are defined using Reactive Action Packages (RAPs) [21] 3T (and similar approaches) encounters problems when models become inconsistent with the true environmental state.

Probabilistic approaches have been developed to help address this shortcoming (see Section 2.6). The work in this thesis falls into this category, but draws from previous approaches. Of note, an arbitration mechanism (similar to that of subsumption) is used to enable or suppress different planner outputs within the hierarchy. Sensory feedback is used online to adapt plans as required by environmental contexts similar to reactive planning techniques.

### 2.6 Belief Space Planning

In general, robot systems operate under partially observable conditions which introduces uncertainty. The complete state required to make control decisions is not available to any single sensor geometry. In fully observable systems, state estimation is fully determined for each sensor geometry and the underlying state space is assumed to be Markovian. Decision making in partially observable systems is properly described as a Partially Observable Markov Decision Process (POMDP). A POMDP [39] is a six-tuple $<S, A, T, R, \Omega, O>$ where: $S$ is a set of states, $A$ is a set of actions, $T$ is a conditional transition probability function between states $P(s'|s,a)$, $R : S \times A \rightarrow \mathbb{R}$ is a reward function, $\Omega$ is a set of observations with elements $z \in \Omega$, and $O$ is an observation function, $P(z|s')$.

In a POMDP, perceptual states cannot be completely differentiated using current observations alone. Memory over a history of actions and observations are required and in some cases, the problem of perception (or acting on percepts) becomes much harder. Papadimitriou and Tsitsiklis proved that finding exact optimal solutions to POMDP problems are PSPACE-complete and thus, intractable for interesting/realistic systems [56]. A common approach to approximating solutions
to POMDPs at runtime are belief space planning approaches, which transform a POMDP to an MDP in belief space [39]. As a consequence, the full range of techniques for solving MDPs can be applied. Although the newly formed belief state is fully observable, it is defined over continuous space and thus infinite.

To find solutions in a belief space in spite of the state-space explosion, several methods have been proposed. Roy proposed a belief-compression algorithm [62] that allows planning to be performed in a lower-dimensional belief space than the original belief space. Maximum likelihood approaches maintain distributions over state but act greedily on the most likely underlying state [58, 66]. Sampling based techniques have been leveraged to explore belief space efficiently [28, 9].

Belief space planning approaches generally combine information gathering with belief condensation to states that solve a task. Heuristic techniques have been used to select actions that address the task while minimizing the impact of uncertainty [69]. “Dual-control” techniques use actions that explicitly reduce uncertainty and actions that explicitly maximize reward [10]. These approaches work well in settings with state-dependent rewards. In settings where reward is dependent on condensing belief rather than a single state $\rho$POMDPs have been proposed [2]. Instead of rewards dependent on a particular state-action pair (as is common in POMDPs) $\rho$POMDPs use rewards related to properties of the belief distribution itself.

### 2.6.1 Active Perception and Active Belief

In order to actively improve belief in perceptual feedback, Aloimonos [1] and Bajcsy [4] introduced the general framework of active perception which is receiving increasing attention in robotics [63, 11]. Denzler et al. demonstrate how using an information theoretic approach to actively decide optimal sensor configurations can improve detection rates [18]. Similarly, Eidenberger and Scharinger demonstrate an active perception approach to improve object detection while overcoming occlusion
through the use of a greedy information theoretic approach that can be used to successfully recognize multiple objects in a cluttered scene [20].

While most studies in active perception do not consider the use of manipulation, some results exist in the literature that incorporate both vision and manipulation to recognize objects. Hsiao et al. use a decision theoretic solution to a POMDP to determine relative pose of a known object. They show that rolling out belief states by just two plies can lead to a drastic decrease in the number of actions, but that planning multiple plies into the future quickly becomes computationally prohibitive [34]. Högman et al. use the action-effect relation to categorize and classify objects [32]. Browatzki et al. [8] use a similar action selection metric and transition probabilities on a view sphere but only employ in-hand rotations that change viewpoints between visual keyframes recorded in a model.

Inspired by these works, Ruiken et al. proposed an Active Belief Planner (ABP) [66] that relies on models called Aspect Transition Graphs (ATGs) [68] to approximate belief transition dynamics. Aspect Graphs were originally introduced in the 70’s to represent an object’s appearance using multiple viewpoints [44] inspired by the human vision system [77]. Aspects in ATGs (known as aspect-nodes) generalize this concept to multi-modal sensor data (visual and tactile). They encode constellations of probabilistic, multi-modal features from specific relative sensor geometries. The transition dynamics of an ATG describe how an agent can interact with an object using stochastic actions to transition between these aspect-nodes. Encoding robot-object interactions this way allows perceptual systems to leverage the history of partial observations to improve discriminative abilities. A partial ATG for a toy block is shown in Figure 2.3. These models support an efficient means of predicting the likely impact of a sequence of actions on a task using histories of actions and observations. ATGs are constructed from extensive, cumulative experience with an object under controlled conditions [79, 48].
Figure 2.3: A partial Aspect Transition Graph (ATG) of a toy block. Constellations of visual features present on the block (the numbers on each face) from given sensory configurations define each Aspect node. The model captures the stochastic nature of outcomes of two actions from the Aspect node on the left, probabilistically leading to different outcome states.
Consider the Active Belief Planner shown in Algorithm 2. Given a transition model $T$ (such as an ATG), the algorithm takes actions $a \in A$, to optimize a reward function $r$ based on $b$, the distribution of belief over the underlying POMDP states $S$. Every execution cycle, the planner computes the next action $a^*$ that maximizes $r$ to a fixed search depth.

$$a^* = \arg \max_{a \in A} r(b, a)$$

To compute $a^*$, all actions are considered given the models and $b_k$, the distribution of belief over states at time $k$. This computation considers how each action $a_k$ will impact $b_{k+1}$. The posterior after the control update $\bar{b}_{k+1}$ is computed given the belief of the current state $b_k$ and the transition probability $Pr(s_{k+1}|a_k, s_k)$ from $T$ (Lines 5-6). Given an expected observation from a forward model through $\text{SimulateObservation}(s_{k+1})$ and an observation function $Pr(z_{k+1}|s_{k+1})$, the expected posteriors are computed (Lines 8-10). These posteriors are then used to evaluate the metric $r$ (Line 12). The action $a_k$ that maximizes $r$ is then chosen to be executed (Line 13). A search depth of one is shown. To plan multiple plies into the future, the planner is simply called iteratively to the prescribed search depth using the previous posterior belief as the new prior.

Sen demonstrated how such a system can scale to large model-sets containing up to 10,000 objects while maintaining reasonable planning times [68]. This was achieved by pruning objects from the current state whose belief fell below a threshold. Although this allows for more tractable planning problems, it has the negative effect of eliminating the potential of recovering the true belief in case of sensorimotor failures. Ruiken et al. addressed this shortcoming by terminating roll-outs of the belief that fell below a threshold rather than purging them from the current belief state entirely [66]. With this approach, the ABP generates plans that monotonically reduce uncertainty in the current state.
Algorithm 2 Active Belief Planner: A planner that optimizes a reward $r$ based on the expected belief state $b_{k+1}$ over $S$ given $a_k \in A$ using a transition function $T$ to select actions. An optional target distribution $G$ can specify a goal distribution toward which to drive the system. $\eta$ is a normalization constant, $O$ is an observation function, and $z$ is a simulated observation from a forward model.

1: function ABP($S, A, T, O, r, b_k, G$)
2:    scores $\leftarrow \emptyset$
3:    for all $a_k \in A$ do
4:        $u(a_k) \leftarrow 0$
5:        for all $s_{k+1}$ do
6:            $\overline{b}(s_{k+1}) = \sum_{s_k} P_r(s_{k+1}|a_k, s_k)b(s_k)$
7:        for all $s_{k+1}$ do
8:            $z_{k+1} \leftarrow \text{SIMULATEOBSERVATION}(s_{k+1})$
9:        for all $s_{k+1}$ do
10:           $b_{k+1}(s_{k+1}) = \eta P_r(z_{k+1}|s_{k+1})\overline{b}(s_{k+1})$
11:        $u(a_k) \leftarrow u(a_k) + \overline{b}(s_{k+1})r(b_{k+1}, a_k, G)$
12:    scores.append($u(a_k)$)
13: return $\arg \max_{a_t}(\text{scores})$

2.7 Experimental Apparatus

The uBot-6 mobile manipulator [64] (shown in Figure 2.4) is used as the experimental platform used throughout this dissertation. uBot-6 is a toddler-sized bimanual mobile manipulator. It balances dynamically on two wheels through an on-board linear-quadratic regulator (LQR) with feedback from a 9 DOF inertial measurement unit (IMU). It has 13 total degrees of freedom (DOF), including two 4 DOF arms and a rotatable trunk which provide a large bimanual workspace. Visual and depth data are provided by an ASUS Xtion RGB-D camera located on a coupled 2 DOF head. In addition to proprioceptive data at each actuated joint, the robot has one six axis force-torque sensor in each hand that provides haptic feedback. Software for the robot is developed within the ROS ecosystem [59].

Closed loop controllers enable the robot to perform manipulation tasks with large objects. These controllers are based upon the control basis framework [35], which is a landscape of attractors defined by parameters describing the robot’s percep-
tual and motor resources and that includes co-articulation using null space control compositions. Using such an approach, parameterized controllers provide robust, sensory-driven interactions with the environment. Motion constraints in navigation are resolved through the use of a harmonic function motion planner [16].

2.7.1 Experimental Objects: ARcubes

As the uBot-6 is best suited for manipulation on large objects experiments we leverage the ARcube domain introduced by Ruiken et al. [66]. ARcubes (See figure 2.5) are rigid cubes whose size can be adjusted to meet the requirements of task specifications. Each of the six faces of the cube is marked with a single ARtag.

Although fiducials like ARtags are usually employed to eliminate uncertainty and make objects fully observable in recognition problems, ARcubes are only partially observable from any single sensor geometry. This is because a particular cube is defined by a specific arrangement of a subset of tags. The natural sparseness of features on any one cube from any one vantage point often leads to a large degree of ambiguity.
Figure 2.5: Two views of ARcubes illustrating the sparseness of features present from different views. On the left (a), only 2 features are visible, while on the right (b), only 3 features are visible. This sparseness of features leads to a large degree of ambiguity within a model-set of ARcubes as sub-sets of features can be shared across multiple objects.

with regards to object identity. Additionally, ARcubes can possess eccentric mass distributions, which lead to different transition dynamics under manipulation. Visually identical objects within a model-set can possess differing mass distributions—leading to different interaction dynamics. This hidden state can only be observed through interaction—it cannot be inferred from vision alone.

ARcubes provide an experimental framework with a controllable range of ambiguity. Sen performed active recognition experiments using model-sets containing up to 10,000 ARcubes with controllable degrees of similarity, thus, producing a well-controlled experimental basis for considering a range of inter-object ambiguity in a fundamentally partially-observable perceptual space [68]. Experimental recognition tasks constructed with these objects are very challenging, requiring multiple interactions with the object as well as the history of observations to fully disambiguate objects within a test-set. Furthermore, the efficiency of the sensing strategy varies based on the composition of the model set.
CHAPTER 3
HYBRID PLANNING GROUNDED IN BELIEF

Using the ABP, Ruiken et al. demonstrated how tasks (such as orienting to an object with particular features) can be addressed using partitions of an ABP’s state-space [65]. Rather than compute metrics and measures such as Information Gain (see Section 2.1), over the entire state-space, they were computed over partitions of the space, where the partitions consisted of objects that satisfied task roles and those that did not. Although tasks such as recognition or orienting to an object are easily specified this way, tasks traditionally encountered in robotics such as re-positioning objects in the environment are not easily addressed.

To leverage the results of the ABP to solve tasks we followed techniques commonly used in task and motion planning that decompose the problem into two interacting and complimentary planning problems—a high-level task planner and a low-level motion planner. High level planners typically use symbolic planners to solve task level problems while motion planners interact with the environment (see 2.2 for more details). Instead of using a motion planner to directly interact with the environment, we use the ABP to manage interactions with a partially observable environment. This approach is similar to Hadfield-Menell et al. as it uses belief space techniques to manage interaction with the environment and a symbolic planner to address task-level problems [27]. This hybrid approach combining symbolic and belief space approaches grounded symbolic abstractions in belief distributions to achieve reliable interactions. This approach is used as a baseline for comparison later in the dissertation.
3.1 Hybrid Task Planner

The hybrid task planner consists of two coupled planning algorithms: a model-based belief space planner (the ABP) and a standard symbolic planner (Fast Downward [31]). The model-based belief space planner is used to overcome uncertainty in lower-level interactions with objects in the environment and to ground symbols—i.e. asserting the symbols required in symbolic planners. The symbolic planner is used at the high-level to handle the resource and geometric constraints of the task. In this chapter, we focus on a task where a robot constructs a copy of a demonstrated arrangement of features. We define an assembly to consist of observable features present on objects from a pre-specified viewpoint. To accomplish this task, the robot must:

1. Identify the target objects required for the assembly
2. Orient the target objects for placement in the assembly
3. Pick and place the target object into the assembly.

The hybrid system can accomplish this task by constructing a symbolic world state based upon interactions with the environment through the ABP. When confidence in ABP state reaches a belief threshold $\beta_{symbol}$, the robot extends its symbolic state to include the newly “discovered” information. This state is then used by the symbolic planner in an attempt to find a solution. If no solution is found, the robot continues to interact with the environment through the ABP. If the symbolic planner finds a solution based upon the current state, the robot executes the solution as prescribed. This procedure is outlined in Algorithm 3.

The ABP uses ATG models of ARcubes (as described in Section 2.7) to predict future world states based upon current beliefs and candidate actions. A subset of observable features $f$ provides support for aspect nodes $x$ that could have been generated by objects in the model-set. Actions $a \in A$ cause transitions between the
Algorithm 3 Hybrid Task Planner

1: while assembly is not complete do
2:   solution = SymbolicPlanner(PDDL problem)
3:   if solution does not exist then
4:     Compute $c_{\text{target}}$ using Eqn (3.3)
5:     Select $a(h_k) \in A_{\text{find}}$ from ABP with find($c_{\text{target}}$)
6:     Execute $a(h_k)$
7:     if bel($c_k^{\text{target}}$) > $\beta_{\text{symbol}}$ in Eqn (3.1) then
8:       Symbolize $c_k^{\text{target}}$ and generate PDDL problem
9:     else
10:    Select $a(h_k, c_k^{\text{target}}) \in A_{\text{pick\&place}}$ from solution
11:   Select $a_{\text{orient}}(h_k) \in A_{\text{orient}}$ from ABP with orient($c_k^{\text{target}}$)
12:   Execute $a_{\text{orient}}(h_k)$
13:   if bel($c_k^{\text{target}}$) > $\beta_{\text{orient}}$ in Eqn (3.4) then
14:     Execute $a(h_k, c_k^{\text{target}})$
15:     Observe and generate PDDL problem

aspects nodes, where the transition probabilities are defined as $p(x_i | x_j, a(\theta))$, where $\theta$ are parameters of the action stored in the ATG. A “hypothesis” is a spatially constrained volume in which we maintain distributions of belief over multiple object models. At run-time, the robot may create a number of hypotheses that is not known a priori as it encounters objects in the environment.

3.2 Partitioning the Belief Space

Following Ruiken et al. [65], the ABP can partition the state space of each hypothesis to address tasks in belief space. In this work, the tasks relevant to be addressed are finding and orienting an object. To address these tasks, the model-set of the planner (the underlying state), is partitioned into different classes based upon some desired, observable qualities that are generally defined as sub-sets of features. Belief is updated during execution as per normal ABP updates (see Section 2.6). Belief over the classes of models $C$ is then computed by summing the belief of aspect nodes $x \in C$ that satisfy a given specification. Rather than evaluate reward across the whole state space, actions are then selected by evaluating reward over these partitions.
3.2.1 Finding an Object of the Target Class

Only a subset of an object’s visual and or tactile properties are needed to satisfy task requirements. A class of objects \( c_{\text{target}} \) is defined such that it creates a partition between target and non-target objects. For ARcubes, this can correspond to objects that have particular aspects (constellations of ARtags such as a ‘3’ and ‘0’) and those that do not.

Task \( \text{find}(c_{\text{target}}) \) can be defined as follows,

\[
\exists \ h_k \ \[ \text{bel}(c_k) > \beta_{\text{symbol}} \], \tag{3.1}
\]

where \( h \) represents an object hypothesis in the scene, \( k \) is the ID of the hypothesis, and \( \beta_{\text{symbol}} \) is a belief threshold. Membership in the partition \( c_{\text{target}} \) is defined

\[
c_{\text{target}} = \{ x_i | \exists x_j \exists o_k [ \ p(o_k|x_i) = 1 \land p(o_k|x_j) = 1 \land \mathbb{1}(x_j) = 1 ] \}, \tag{3.2}
\]

where \( o \) is an object, \( x \) is an aspect node of the object, \( p(o|x) \) is the probability that aspect node \( x \) belongs to object \( o \), and \( C \) is a set of aspects that satisfy the task. \( \mathbb{1}(x) \) in an indicator function that evaluates to 1 if \( x \in C \) and 0 otherwise. Information Gain (see Section 2.1) is used as a metric to select the best next action \( a \in A_{\text{find}} \) to reduce the uncertainty of class membership.

To copy a reference structure, the robot may need to find multiple target goals represented as partitions of the object model set. These classes are defined

\[
c_{\text{target}_1}, c_{\text{target}_2}, \ldots, c_{\text{target}_M}
\]

where \( M \) is the number of targets necessary for the task. The robot selects the target, \( c_{\text{target}_i} \), to investigate next by selecting the partition whose entropy (see Section 2.1) is the highest among these target goals \( c_{\text{target}_j} \).
3.2.2 Orienting an object

After an object of the target class $c_{\text{target}}$ is found the robot can orient the object to a specific aspect of the object. This is required to prepare for *pick-and-place* actions that re-position objects in the environment. Orienting to specific aspects during pre-grasp allows the robot to place objects in orientations that satisfy task requirements. A subset of aspect nodes $x$ of hypothesis $h_k$ that satisfy $\text{orient}(c_{\text{target}})$ is defined

$$\text{bel}(c_{\text{target}}^k) > \beta_{\text{orient}} | c_{\text{target}} = \{x_j | 1(x_j) = 1\}.$$  \hspace{1cm} (3.4)

Actions to condense belief in the task partition can be selected by finding the shortest path to aspect-nodes in the partition using the maximum likelihood ATG. If the incorrect ATG is initially assumed, belief updates during execution will drive the distribution towards the correct state.

3.3 Symbolic Planning and Symbol Grounding

Existing symbolic planning techniques are well suited to resolve action preconditions and resource constraints in planning problems. However, in unstructured and partially observable environments the state (symbols) necessary to solve these problems may not be initially available. To solve these problems an agent must use sensory inputs to form the symbols that are needed.

In order for such representations to be reliable we ground the symbols by deriving them from aspects in ATGs. These symbols can then be realized by condensing belief at run-time. By using aspects as symbols, we can aggregate all the aspects (regardless of the object that they belong to) that support the task using task partitions. When the belief across a partition is sufficiently high, we can *symbolize* the target aspect

\[
c_{\text{target}} = \arg \max_j H(c_{\text{target}}). \hspace{1cm} (3.3)
\]
for use in planning. It is assumed that a task domain has been specified in PDDL \textit{a priori}. Inspired by [71], we use an interface layer to translate between the symbolic and belief-based physical representations in order to generate PDDL problems at run-time (Lines 8, 15 in Algorithm 3)(See Figure 3.1 for an example).

```
(define (problem auto-generated)
  (:domain arcube-assembly )
  (:objects floor - aspect h2 - obj
    h0 - obj h1 - obj
    1-x - aspect 3-0 - aspect )
  (:init (clear floor) (valid_object h2)
    (valid_object h0) (valid_object h1)
    (has_aspect h0 1-x) (has_aspect h1 1-x)
    (has_aspect h2 1-x) (has_aspect h2 3-0) )
  (:goal (and (on 1-x floor) (on 3-0 1-x)) ))
```

Figure 3.1: Example of an ARcube assembly problem generated during execution. The :goal statement is generated when the robot registers the demonstrated assembly. The :init and :object statements are updated after each symbolization.

The main symbols used in our system are directly derived from the ATG model set provided to the robot and the specified goal. Additional constraints (such as supporting/on relations) are determined using additional geometrical constraints. After a problem has been generated, it is submitted to a symbolic planner [31] that quickly finds a satisficing plan if one is available. In the problems we considered, satisficing was a more practical choice over optimality due to the inherent partial observability present in the task. If a plan does not exist the robot continues to symbolize its environment until one can be found.

### 3.4 Demonstration Platform and Domain

To demonstrate the proposed mechanism, we conducted a simple copy task with resource constraints using the uBot-6 mobile manipulator and the ARcube objects (see Section 2.7).
ARcube ATGs are created using a mixture of intrinsically motivated structure learning [79] and hand-built models. ARCube ATGs include flip, lift, orbit, push, and drive actions. orbit is a locomotive action in which the robot rotates around the target object to change the viewpoint. flip is a manual action that rotates the target object. push is a manual action in which the robot extends an arm and pushes along a particular normal on the target object. lift is a manual action the robot performs a grasp and lift the object. drive is a locomotive action that leads to a specific aspect node. A Hough transform [5] is used to match raw features to aspects. These aspects are then used as the observation $z$ to update the belief over aspect nodes (and thus belief over ATGs). Belief distributions over multiple object models are maintained over spatially constrained models defined as hypotheses $h$. 

3.4.1 Assemblies

We construct assemblies out of objects in the ARcube model-set. This model-set can be used to precisely control planning complexity. Each ARcube has 48 aspect nodes. Twenty different ARcubes are in the model set in our demonstration, yielding $n \times 20 \times 48$ states for the belief space planner to consider where $n$ is the number of building blocks present. The ABP uses an adaptive search depth as in [65] where belief is propagated further into the future if the next action is planned in $\alpha$ seconds. In this work, $\alpha$ was empirically chosen to be 0.6. In our model set, the first ply of the search tree is only computed in such a short time after belief has been somewhat condensed. This more informed belief state reduces the width of the search tree, making planning additional plies practical. Planning further in the tree better informs action selection.

The robot constructs a copy of a stack that is presented $a$ priori. The copy task requires the proper arrangement of two ARcubes. The robot first needs to register the goal of the task by observing a target stack. There will be two target aspects in the stack: one for the bottom and another for the top. The robot then observes the
The copy task involves using the building blocks displayed in Figure 3.2 to copy the structure. Although not visible from this view, hypothesis $h_2$ has both aspects ‘1-x’ (an aspect with ‘1’ in front) and ‘3-0’ (an aspect with ‘3’ in front and ‘0’ on top), while hypotheses $h_0$ and $h_1$ only have aspect ‘1-x’. This introduces a resource constraint as only hypothesis $h_2$ can be used for the top object in the structure. The robot determines what aspects are missing for the copy, then generates partitions to search for these aspects. There are three ARcubes to consider as building blocks for the replica. Each of the three affords aspects that can be used for the bottom block, but only one of them satisfies the top (see Figure 3.2). The planner must resolve resource constraints before arranging the objects.

Initially, the robot is unaware of the number or identity of building blocks available for the copy task. The number and identity are exposed through the belief space planner. Actions used by the symbolic planner are limited to pick-and-place actions. These symbolic actions are realized through an interface layer of actions $A := \{\text{pick-and-place, orient}\}$, which resolves motion constraints by orienting the objects if needed. Actions used for orient and find tasks are $A_{\text{orient}} = A_{\text{find}} := \{\text{orbit, flip, lift, push}\}$. pick-and-place actions $a \in A_{\text{pick-and-place}}$ take two parameters: which hypothesis to pick and the place goal. The parameters for symbolization $\beta_{\text{symbol}}$ and orientation $\beta_{\text{orient}}$ are set to 0.9 and 0.7 respectively in the demonstration.
3.5 Outcomes

The robot successfully constructed the copy as shown in Figure 3.3 despite the resource constraint situation. The belief that each hypothesis fulfilled the roles necessary over a task execution is shown in Figure 3.4. The figure shows when hypotheses were symbolized, when targets were oriented, and when pick-and-place actions were started. Executing physical actions dominates planning time. Physical actions took 38.8 seconds on average. On average, ABP with 1-ply took 0.26 seconds, and 3.7 seconds for 2-ply. The symbolic planner took 0.23 seconds on average. The efficient computation is partially due to the simple domain and problem we are addressing at the symbolic level.

In our demonstration the robot correctly symbolized the objects needed to complete the task as seen in Figure 3.4 while not falsely symbolizing improper objects even though symbolizing objects in partially observable problems (such as those with ARcubes) is difficult. Symbols should be generated only when the semantics of actions are vetted by the belief space planner. To illustrate, the result would have been different if we had used a maximum likelihood observation to symbolize after the first action at 423 seconds (Figure 3.4). The plan formed from this belief would have led
Figure 3.4: The figures show the task beliefs for each hypothesis in the scene for finding aspect ‘1-x’ (top figure) and aspect ‘3-0’ (bottom figure). The robot symbolized all of the hypothesis with aspect ‘1-x’ after the first observation at 423 seconds, and symbolized hypothesis $h_2$ with aspect ‘3-0’ at 759 seconds.

We performed ten different demonstrations changing the number and identity of building blocks present in the scene that could be used in the copy task from two to four for the two object stack problem. In these demonstrations, eight successfully symbolized and planned a solution. The two failures were due to unexpected outcomes produced by stochastic actions. These errors were not anticipated or managed by the symbolic planner.
3.6 Discussion

In this chapter, a hybrid planning architecture that uses a model-based belief space planner and a symbolic planner was presented. With this approach, a belief space planner stabilizes the semantics of feedback from the environment through interaction to enable reliable symbolization of higher level abstractions. Results were shown for this hybrid architecture on a real robot system performing a two-block stacking assembly with resource constraints. Thanks to the symbolization grounded in belief, the system was able to overcome the resource constraint in examples that would have prevented maximum likelihood approaches from succeeding. However, failures did occur when symbolic dynamics did not match real-world dynamics. In the next chapter, an approach to overcome this limitation by using a hierarchy of belief space planners is proposed.
CHAPTER 4

HIERARCHICAL BELIEF SPACE PLANNING

This chapter introduces a hierarchical belief space approach to better manage uncertainty at run-time at different levels of abstraction. The approach relies on representing a task hierarchically, using different levels of description, and implementing layers of active belief to cause useful, multi-level transitions to shore up multi-level belief that a task specification has been achieved. By managing belief distributions over multi-level abstractions, we investigate how much a particular robot-object interaction will contribute to the task. A hierarchy of planners is used to improve tractability in problems that face robots. If instead a “flat” planner is used, the state space of the problem quickly becomes unmanageable for planning systems—preventing them from providing meaningful guidance at run-time within reasonable time-frames (i.e. planning times of seconds, not minutes). This problem is often referred to as the curse of dimensionality—the number of future states that must be considered increases exponentially in the dimensionality of the state space. Planning in hierarchical systems helps mitigate this by reducing the combinatorics of the planning problem.

Hierarchical approaches to address POMDPs have been previously investigated [30, 40], but generally reason over hierarchies of actions—that is they organize actions hierarchically to reduce planning time within a single planner. Foka et al. investigated using hierarchies of action and state in a navigation domain [22] where the robot does not actively alter the environment. In this work, we leverage hierarchies of planners to reduce uncertainty at many levels of abstraction in a general mobile manipulation context using multi-modal feedback. Sridharan et al. use a hierarchical formulation
but only rely on actions that expose new information [70]. In contrast, the proposed approach considers actions that are both informational and functional—that is they expose new information and accomplish task objectives.

### 4.1 Hierarchical Belief Space Planning

As defined in Section 2.6, a single instance of an ABP is defined by the tuple \( < S, A, T, O, r, b_k, G > \). In a hierarchy of \( d \) planners an extension is required to allow higher levels to leverage information obtained at lower levels. To this end, we introduce a state abstraction function \( Z \) which generates observations from lower levels of the hierarchy. This is achieved by first sampling the belief state of the \( i - 1 \) level at time \( k \),

\[
\hat{b}_{i-1,k} \sim b_{i-1,k}.
\]

The state abstraction function \( Z \) transforms this sampled belief \( \hat{b}_{i-1,k} \) to an appropriate form to be used as an level \( i \) observation \( z_i \) at time \( k \).

\[
z_{i,k} = Z_i(\hat{b}_{i-1,k})
\]

For succinctness, we will use \( Z_i(b_{i-1}) \rightarrow z_i \) to represent this process. This transformation is necessary as the different belief spaces may reason in different underlying state spaces. Sampling lower-level belief distributions allows information to be transferred effectively across different state spaces in the hierarchy. The hierarchical form of the ABP, \( Hierarchical-ABP \) (HABP) is thus defined by the set of individual ABPs and their state abstraction functions \( Z \).

#### 4.1.1 Example State Abstraction Function

Consider two belief spaces, \( B_1 \), a categorical distribution over object identity, and \( B_2 \), a continuous distribution over object position in the environment. For simplicity, assume the underlying state space of \( B_1 \) consists of
ATG models. Belief in $B_1$ can be used to form observations for the second belief space through the state abstraction function $Z_2$. The belief $b_1$ can be sampled to obtain $\hat{b}_1$.

$$\hat{b}_1 \sim b_1 \quad (4.1)$$

$\hat{b}_1$, which could be obtained through maximum likelihood or weighted sampling techniques, latently encodes the pose of the modeled object. This pose can then be used as an observation $z_2$ and can be used to update the belief $b_2$.

The $i^{th}$ planner in a hierarchy of ABPs of depth $d$ is defined: with $S_i$, the set of world states, $T_i$, the conditional transition probability between states, $O_i$, an observation function, $A_i$, the set of available actions, $b_k$, the belief distribution over states $\in S$ at time $k$, $r_i(b_i, A_i) \rightarrow \mathbb{R}$, a reward function parameterized by the belief distribution and actions, and $Z_i$, a state abstraction function $Z_i(b_{i-1}) \rightarrow z_i$.

For $i = 0$, a distribution of feature positions and covariances $f_0$ is computed by a perceptual front-end (in place of $b_0$). The state abstraction function allows each successive layer of the hierarchy to form observations based on the belief state of the preceding layer. This creates high-level belief distributions that have been stabilized by the lower levels of the hierarchy. This data flow is shown in Figure 4.1. Each time step, the hierarchy is updated from the bottom-up and new observations are fused with the existing state. This fused state is used to plan $n$-ply forward into the future using $T_i$ and $A_i$. Future observations $z_{k+[1...n]}$ are estimated using the forward model. This process is outlined in Algorithm 4 and shown graphically for an arbitrary level of a hierarchy in Figure 4.2.

This approach relies on implementing layers of active belief to cause useful, multi-level transitions to shore up belief that a task specification has been achieved. By managing belief distributions over multi-level abstractions, we investigate how much a particular robot-object interaction will contribute to the task. Each planner in the
Figure 4.1: Graphical representation of data flow in a hierarchy of $n$ levels. Belief from the lower levels is used as inputs in higher level planners. Each level of the hierarchy $j$ selects actions $a_j^*$ to execute. An arbitration module named the Arbiter selects an action $a^*$ to execute from the set of prescribed actions.
Figure 4.2: Graphical representation of an arbitrary level of the hierarchy. Belief from the lower level and the previous time step state of this level are fused into a current state estimate (left side). This state estimate is used to plan forward into the future using $T_i$ and $A_i$ (right side).
Algorithm 4 HABP : Algorithm for updating a hierarchy of depth $d$ at time $k$ and selecting the next action to execute.

1: function HABP($f_{0,k}$)
2:     $b_{0,k} \leftarrow f_{0,k}$
3: for $i$ in range 1 to $d$ do
4:     $b_{i,k} \leftarrow \text{Bayes}(b_{i,k-1}, a_{i,k-1}, Z_i(b_{i-1,k}))$
5: for $i$ in range $d$ to 1 do
6:     $a_i^* \leftarrow \text{ABP}(S_i, A_i, T_i, O_i, b_{i,k}, r_i)$
7:     if Arbiter ($a_i^*$) then
8:         return $a_i^*$

hierarchy selects actions $a^*$ that maximize the reward $r_i$ at level $i$ given actions in set $A_i$ and the current belief $b_i$

$$a^* = \arg \max_{a \in A_i} r_i(b_i, a).$$

Task-level reward $r_d$ depends on the confidence in lower-level abstractions. If the entropy of the distribution over belief $b_{d-1}$ is high, it will support many different possible observations $z_d$ and, therefore, provide little new information or guidance to the planner at level $d$. Actions $a \in A_{d-1}$ can improve the precision of the state and, thus, enhance the performance on the task.

4.2 Confidence-based Subsumption in Belief Space

In a hierarchy of planners with limited sensor and effector resources a decision regarding control authority must be made at each control step to determine which planner’s prescribed action will be executed in order to optimize performance of the task. This arbitration is accomplished through an Arbiter module. Many different strategies exist for coordinating interactions between multiple ABP layers and the external environment that respect the hierarchical description of the task. To effectively make these decisions, we draw upon ideas from reactive control, particularly
subsumption architectures to select actions that will improve the robot’s confidence that task specifications have been achieved.

In a subsumption architecture, a hierarchy of controllers are combined to form complex behaviors. Higher-level controllers can inhibit or allow lower level control outputs dependent on current state and sensory information [7]. In the hierarchy of active belief planners decisions at the higher levels are only informative given that lower level beliefs are confident. The highest-level planner with enough confidence in its state is chosen for execution. This limit can be set arbitrarily high. In this way, belief actively condenses in the hierarchy from the bottom-up. Directing actions into the lowest levels where minimum confidence levels have not yet been established and then advancing results in a reasonable, conservative strategy.

![Diagram](image)

Figure 4.3: Graphical representation of the confidence-based subsumption Arbiter for a hierarchy of arbitrary depth $n$. When enough confidence exists in higher levels they suppress the outputs of lower-level planners.

This is analogous to the reactive nature of subsumption—when enough confidence in state exists at a particular level, lower-level outputs are inhibited and the action chosen at that level is executed by the robot. If a high-level planner loses confidence it
may yield execution to lower-level planners, which in turn can improve the confidence in state at the higher-levels. It should be noted that the output from each level is only computed after updating through its respective state abstraction function. This way, if confidence is lost in lower level abstractions the higher-level state reflects this. This is the approach investigated in this work. An example implementation of a two-level hierarchy is described in the following section.

### 4.3 A Two-Level Hierarchy for Assembly

The assembly domain uses objects known as ARcubes introduced in [65]. ARcubes are rigid cubes whose size can be adjusted to meet the requirements of the task. Each of the six faces of the cube is marked with a single ARtag\(^1\). The natural sparseness of features on any one cube leads to a large degree of ambiguity with respect to a set of ARcubes. We use Kalman Filters to track ARtag positions in $\mathbb{R}^3$.

Recent work has started to use ARcubes to form simple assemblies such as towers. In that work, a symbolic planner manipulated symbols grounded in belief by a belief space planner to resolve action pre-conditions and resource constraints in unstructured environments [73]. This system demonstrated how symbols grounded in belief lead to more reliable solutions than using maximum likelihood assumptions alone. However, assembly-level actions were not risk compensated. As a result, unexpected outcomes occurred when objects were not placed precisely in the assembly. Without pro-active management of uncertainty or special purpose recovery mechanisms, such outcomes require external resets of the system.

In this example domain, we use a two-level hierarchy. Belief over the assembly state is managed in the top level of the hierarchy and belief over objects is managed at the lower level. Each control step, the belief at each level in the hierarchy is updated

\(^1\)https://artoolkit.org/
and each layer plans \( n \)-plies into the future after which an Arbiter selects an a
sequence of actions to execute (as per Algorithm 4). The first action in this \( n \)-action
sequence is executed, after which the hierarchy re-plans with updated state.

### 4.3.1 Object Level

The bottom of the two-level hierarchy manages noisy interactions with the en-
vvironment. ARcube ATGs are used as forward models. Given a model-set \( M \), the
ATG for object \( m \in M \) is a directed multi-graph \( G_m = (X_m, U_m) \) where \( X_m \) is a set
of aspect nodes and \( U_m \) is a set of actions that represent edges in the graph. Edges
encode the transition probability \( T_1 \) between states. An aspect node \( s \in X \) consists
of a set of features, \( s = (f_{obj}^1, f_{obj}^2, \ldots) \), that can be observed from a particular sensor
configuration. A feature \( f_{obj} \) is a tuple of feature id, \( \xi \), and its location in the object
frame described using a Gaussian distribution, \( f_{obj}^i = (\xi, \mathcal{N}(\mu, \Sigma)) \) where \( \mu \in \mathbb{R}^3 \) and
\( \Sigma \in \mathbb{R}^{3 \times 3} \). During the task execution, the robot keeps updating belief distributions
over aspect nodes \( s \) for each hypothesis. A “hypothesis” is a spatially constrained
volume in which distributions of belief over multiple object models are maintained. \( b_1 \)
are belief distributions over ARcube ATG aspect nodes, \( b_1 = [b(s_0), b(s_1), \ldots, b(s_{|S_1|})] \)
where \( S_1 = \bigcup_{m=1}^{|M|} X_m \). Each aspect node defines an object frame for its parent ATG.

ARcube ATGs include parameterized mobility and manipulation actions that pop-
ulate \( A_1 \). Mobility actions reconfigure the robot’s relative orientation to an object.
Manipulation actions reconfigure the object relative to the robot through prehensile
actions or non-prehensile pushing actions. Manipulation actions may or may not cre-
ate in-hand rotations that re-orient the object (depending on the action parameters
and object identity). By discretizing these parameter spaces we can define 4 prehen-
sile and 1 non-prehensile manipulation actions and 7 mobility actions. When selecting
actions, the maximum likelihood object frame is used to parameterize actions. Ac-
tions at level 1 accrue belief over ATGs that supports classification and recognition
over the history of observations. Observations $z_1$ are produced with the state abstraction function $Z_0(\{f_1 \cdots f_n\}) \rightarrow z_1$, which turns environmental features into candidate aspects. To perform this transformation a generalized Hough transform is used. This scores candidate aspects from the ATG model-set, and forms a belief distribution over candidate aspect-nodes [66]. Using these observations, the planner can reason about how to manage uncertainty in this level’s belief distributions.

We will use task partitions similar to Ruiken et al. [65] to encode find tasks for the robot at the object level. Belief that a hypothesis $h$ belongs to a target task partition $c_{\text{target}}$ with goal features $s_j$ exceeds belief threshold $\beta$, is defined as

$$b(c_{\text{target}}) > \beta | c_{\text{target}} = \{s_j | \mathbb{1}(s_j) = 1\}. \quad (4.2)$$

where $c_{\text{target}}$ is the target partition—the subset of objects in the model space that satisfy task requirements—and $\mathbb{1}(s_j)$ is an indicator function that evaluates whether $s_j \in c_{\text{target}}$.

Find tasks drive the robot to reduce uncertainty among object models that do and do not support a task. These classes are defined by the goal of an assembly, e.g. cubes with a ‘2’ feature and cubes without. By using Information Gain ($IG$) as $r_1$, the robot can take actions to condense belief on subsets of the model-set effectively. $IG$ in this setting is defined as:

$$IG = H(s_k) - \mathbb{E}_{s_{k+1}}[H(s_{k+1}|a_k)] \quad (4.3)$$

where $s_k \in b_k$ and $s_{k+1} \in b_{k+1}$.

4.3.2 Assembly Level

In the top level of the hierarchy, uncertainty in the spatial precision of the assembly is managed. This is achieved by maintaining belief distributions over positions of
goal features in the environment. This defines $S_2 \in \mathbb{R}^3$. In this implementation, $Z_2$ samples the maximum likelihood state of $b_1$ to “observe” the expected positions of these features on objects in the environment. This creates observations $z_2 = \{p_1, \cdots, p_h\}$ where $p_i \sim \mathcal{N}(\mu, \Sigma)$, with $\mu \in \mathbb{R}^3$, the mean position and $\Sigma \in \mathbb{R}^{3 \times 3}$, reflecting positional uncertainty of these features on the maximum likelihood object of each of $h$ hypotheses in the environment. $Z_2$ only allows object belief to be lifted to this level if they exceed a belief threshold (80%).

At the assembly level, actions $a \in A_2$ consist of actions that orient and pick-and-place objects in the environment based upon the maximum likelihood ATG. This is achieved by solving a shortest path problem in this ATG using the negative log of the transition probability as cost. Additionally two special actions: a NOP and FIND action are included in $A_2$. NOP allows the robot to stop if it believes the task has been completed. FIND is returned if not enough state exists in $b_1$ to solve the task. A forward model reasons over the geometric effects of actions $a \in A_2$ given $b_2$ and provides $Pr(s_{2,k+1}|s_{2,k},a_{2,k})$, the transition probability $T_2$ of these actions. Given $z_2$, we compute the observation probability $Pr(z_{2,k+1}|s_{2,k+1})$ with empirical models of robot performance of pick-and-place actions $a \in A_2$. We assume these models are Gaussian $\mathcal{N}(\mu, \Sigma)$. The continuous state introduces minor changes to Algorithm 2, which assumes discrete state. In particular, the belief update (Lines 5-10) is computed with

$$b(s_{2,k+1}) = \eta Pr(z_{2,k+1}|s_{2,k+1}) \int_{s_{2,k}} Pr(s_{2,k+1}|s_{2,k},a_{2,k})b(s_{2,k}), \quad (4.4)$$

using the same variables as Algorithm 2.

To quantify performance at the assembly level, we reason over how actions will decrease the Kullback-Leibler divergence $D_{KL}$ [50] between the belief distribution $b_{2,k}$ and a goal distribution $G$. $G$ defines the goal positions of target features in the assembly $(f_{1}\text{goal}, f_{2}\text{goal}, \cdots)$ and specifies the amount of acceptable uncertainty at the goal given empirical models of the robot’s performance $\mathcal{N}(\mu, \Sigma)$. To measure this
divergence, we use error distributions $E$ computed using the $\ell_2$ distance between goal feature positions and feature positions given the belief distribution $b_{2,n}$ given action $a_{2,k}$ and $G$. This yields the following reward function $r_2$

$$r_2 = -D_{KL}(E \parallel G) = -\int_{-\infty}^{\infty} e(x)\log\left(\frac{e(x)}{g(x)}\right)dx$$

(4.5)

where $e$ and $g$ represent the densities of $E$ and $G$ respectively. When the $D_{KL}$ between the current belief state and the goal is less than the $D_{KL}$ between the expected belief state at time $k + l$ ($l \in [0, n]$) and the goal, the planner selects $NOP$ to indicate that the task has been condensed as much as possible (with regards to the performance models of the robot). To bias the robot towards faster solutions (at the cost of precision), the reward function is penalized by the cost of actions. In this implementation, pick and place actions have a uniform cost of $\lambda = 1.0$, yielding the following reward function,

$$r_2 = -D_{KL}(E \parallel G) - \sum_{i=1}^{k+l} \lambda_i.$$  

(4.6)

4.3.3 Control Authority: Arbiter

As stated earlier, control authority will be given to the highest level of the hierarchy where confidence has been achieved. Every time-step both the assembly and object level planners plan the next action to be executed. If the assembly-level planner does not contain enough confidence in state to plan into the future, it returns $FIND$, which yields authority to the object-level planner. If the assembly-level planner returns $NOP$ the system stops execution as the robot has condensed belief to a solution of the task.

This control structure enables an intrinsically lazy behavior for assembly tasks. If not enough state is present in the assembly level planner, then the object level planner is leveraged to uncover the needed state from the world. By relying on the assembly
level when enough state exists to find a solution to the task the robot attends to
the minimum amount of world state required to solve a task. This helps increase
tractability and reduce the dimensionality of problems—as the robot will not actively
uncover state on objects that are not required by the task.

4.4 Example Application: Simple Assemblies

To highlight the approach, we conducted three experiments in the simple assem-
blies domain. Assemblies are specified by configurations of ARcube features in specific
positions. The robot is provided with a model-set of 20 ARcubes modeled as ATGs.
In each experiment, the raw materials (ARcubes) needed to construct the assembly
are present and (partially) observable to the robot, although their poses and identities
are initially unknown to the robot. The first experiment compared the overall error
at the conclusion of a tower assembly using the proposed method to a baseline system
used in [73]. This baseline provides a null hypothesis for evaluating the impact of the
hierarchical system. The baseline system uses a single-level ABP to reason over ob-
ject identities and then lifts objects in the environment to symbolic representations.
These symbols are then used to execute a plan prescribed by a symbolic planner. This
decomposition of the problem is typically used in task and motion planning problems
(see Section 2.2). Committing to symbolic representations (prematurely) will lead to
brittle, error-prone behavior when uncertainty compromises the presumed semantics
of the world. The second experiment demonstrates the flexibility of the approach by
constructing a pyramid of blocks. This assembly challenges the system as previously
placed blocks can be disturbed when placing additional blocks. The third experi-
ment highlights the system’s ability to overcome uncertainty induced by an external
source. This is demonstrated by having a researcher intentionally alter the outcome
of an action during a pyramid assembly.
Closed loop controllers are used in all actions, based upon expectations from the currently observed belief state. These controllers achieve their objectives by following gradients in a convex potential function $\phi(\sigma)$ with respect to changes in the value of the motor resources where $\sigma$ are sensory inputs [13]. It should be noted that these expectations will not necessarily exactly match actual action outcomes as this is a real system. Convergence of these controllers does not imply completion of tasks since the goal of the potential field is impacted by noise from sensory inputs (joint angles, feature locations, etc...) and localization errors. For example, if we have 0.03 m in localization error, the controllers may converge to a goal offset 0.03 m from the ground truth goal. It is not possible to know this offset at run-time. In ideal environments (such as in simulation) we may not have these issues. Our system aims to minimize the impact of these unavoidable sources of noise in the real world by considering the belief of states in addition to minimizing errors in controller and perceptual units.

For placing actions, 120 trials were conducted by the robot placing objects from the ARcube model-set yielding a gaussian distribution with

$$\mu = (0.03345256, -0.04095612) \text{ (m)}$$

and

$$\Sigma = \begin{bmatrix} 0.00324572 & 0.00073537 \\ 0.00073537 & 0.00235752 \end{bmatrix} \text{ (m}^2).$$

This distribution is used to evaluate sampled future states and to compute the observation function, $Pr(z_{2,t+1}|s_{2,t+1})$ in Algorithm 2.

### 4.5 Experimental Results

#### 4.5.1 Comparison to Baseline Planner

The first experiment tasked the robot to construct a tower consisting of two AR-cubes. In this assembly the bottom cube was required to have a “2” tag in front and a
“1” tag on top. The top cube was required to have a “3” tag in front and a “0” tag on top. The features were required to be located in specific positions relative to a fixed frame defined by 3 tags (“A”, “B”, and “C”) as seen in Figure 4.4. We compared the proposed method with a baseline method [73]. When using the baseline, the system would fail when action outcomes did not match the expectations of the planner.

We performed five trials for the hybrid planner and HABP. The robot successfully built the assembly one out of five times with the baseline and five out of five times with HABP. The final assemblies using both approaches are shown in Figure 4.4. The average of the final assembly errors and the results of a T-test are shown in Table 4.1. The p-value of the error is less than 0.05, which shows that HABP has statistically significant better outcomes than the baseline system.

The stochastic nature of action outcomes caused objects to be imprecisely placed in several trials using both methods. If this occurred in the baseline, the robot would often fail to successfully place the second object, causing it to fall to the floor. With HABP, the robot successfully suppresses this error by reasoning about the likelihood of achieving the goal by re-placing poorly placed objects. Using the HABP approach, the robot does not take further actions when the risk of damaging the current assembly with continued interaction is sufficiently high and chooses to maintain the current world state. Risk management is intrinsic to the HABP and does not need to be directly specified or managed externally.
Figure 4.4: Final tower assemblies produced by the two approaches in the first experiment (comparing to a baseline). On the left are the five assemblies using the baseline approach. Due to uncertainty and stochastic actions the assemblies created with this approach often fail. On the right are the outcomes using the HABP framework. By managing uncertainty and risk the planner is able to construct better approximations of the goal assembly.
<table>
<thead>
<tr>
<th>Approach</th>
<th>Final Error ($m$) ± variance ($m^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.349 ± 0.0157</td>
</tr>
<tr>
<td>HABP</td>
<td>0.071 ± 4.05 E−5</td>
</tr>
<tr>
<td>T-test</td>
<td>$p = 0.011 &lt; 0.05$</td>
</tr>
</tbody>
</table>

Table 4.1: The mean ($m$) and variance ($m^2$) of final error ($\ell_2$ distance) in tower experiments (shown in Figure 4.4) using the baseline and the hierarchical framework (HABP). Lower error values indicate better performance. As $p < 0.05$, the results show statistically better performance of HABP approach compared to the baseline.

4.5.2 Pyramid Assembly

The second experiment was designed to demonstrate the flexibility of the approach by constructing a pyramid of three blocks. This assembly requires the two bottom objects to be in close contact with each other to provide enough support for the top block. This close proximity can lead to induced errors during assembly as the first block can be unintentionally disturbed when placing the second block. The goal required the lower left block to have a “4” tag on front and a “2” tag on top, the lower right block to have a “7” tag on front and a “1” tag on top, and the top block to have a “3” tag on top and a “0” tag on top. Goal positions were once again specified relative to a fixed frame defined by 3 tags (“A”, “B”, and “C”).

The only change between these runs and the towers in the previous experiment is the specification of the overall goal—the configurations of the ARcube features in specific positions. Using the proposed approach, the robot is able to construct low-error approximations of the goal assembly, while recovering from induced errors during assembly. This error recovery artifact (similar to recovering from topples in the first experiment), does not need to be specified or managed externally and arises naturally from our approach. Snapshots from the robot’s perspective of assembly progress for a pyramid assembly are shown in Figure 4.5.
Figure 4.5: Example pyramid assembly using the proposed approach. Each picture is a snapshot of the assembly from the robot’s perspective during the task. In between each image the robot places new objects into the assembly or replaces existing objects if it predicts it is capable of achieving a better outcome.
4.5.3 Performance in Face of External Perturbations

Another useful artifact that arises from our approach is best highlighted when a perturbation is intentionally introduced by the experimenter to alter the outcome of the robot’s action. With our approach, the robot recovers from such unexpected outcomes without additional recovery mechanisms. This process is best shown in Figure 4.6. In this demonstration, the robot is constructing a pyramid similar to those in the second experiment. After placing the first object (Figures 4.6a-4.6b), a researcher alters the outcome of a placing action by flipping the box the robot just placed (Figures 4.6c-4.6d). The robot observes this outcome in Figure 4.6e. Due to the unexpected transition, the robot is no longer confident it has satisfied the sub-task (placing the object in the correct orientation). The planner (correctly) selects to flip the cube back to the correct orientation (Figures 4.6f-4.6g). The robot then replaces the object in the correct position in the assembly (Figures 4.6h-4.6k). After observing this satisfactory outcome (Figure 4.6k), the robot then moves to place the remaining objects (Figure 4.6l) and completes the assembly (not shown).

4.6 Discussion

Assembling blocks in unstructured environments provides an ideal domain for investigating the sensitivity of performance to undetected or hidden state. Small errors introduced through stochastic actions early on in an assembly can be magnified as more blocks are placed. By framing the problem in a hierarchical belief space, we can address errors during execution without additional recovery structure. In this chapter, we only considered actions that place objects at goal locations. This excludes actions that disassemble beyond one step. For more general disassembly actions, an extension is required to the framework to generate an additional place goal in each ply of the search-tree to consider the impact a partial (or possibly full) disassembly would
Figure 4.6: Outcomes of intentional perturbation by an experimenter during a pyramid assembly. The robot begins to construct the assembly (a-b). The robot is no longer confident it has placed the object in the correct orientation after the unexpected action outcome (c-e). The robot re-configures the planner to reduce this uncertainty (f-h). After the uncertainty has been addressed, the robot replaces the object in the assembly in the correct orientation (i-k) then moves to complete the rest of the assembly (l).
have on the task. This can be important if the robot (or experimenter) introduces
errors in previously placed objects that are inaccessible from the current state.

Online error recovery is an open problem that must be addressed before robots
can interact reliably with unstructured environments. In complex tasks that employ
tens or hundreds of actions in a sequence, a single fault can disable the robot and/or
damage the environment. In order to embed autonomous robots in unstructured
worlds, it seems reasonable to specify a reliability in excess of thousands of control
decisions between failures, where a “failure” denotes a situation that requires external
reset. To meet these goals, breakthroughs are required concerning the assessment of
uncertainty—specifically as it puts a task at risk—and in the formulation of risk
averse and error recovery behavior.

The hierarchical belief space framework contributed in this chapter meets these
goals. The experiments and demonstrations in this chapter demonstrate that the
robot can recover from errors at run-time without additional task or recovery struc-
ture by continuing to select actions that condense belief to goal distributions. By
structuring the task hierarchically, the robot is able to reason and address errors at
multiple levels of abstraction—at the level of object or assembly—to support more
robust autonomous robots.
CHAPTER 5
LONG-TERM AUTONOMY UNDER UNCERTAINTY

Cleaning up our laboratory (stowing experimental objects, replacing/emptying trash receptacles, pushing chairs back under desks, etc.) is representative of a class of tasks that require long-term management of uncertainty and risk. Such a task involves on the order of hundreds of autonomous control decisions without external resets in an unstructured environment. A common approach used in robotics to complete such tasks involves task and motion planning (see Section 2.2). These approaches generally decouple task planning from motion planning and address each as separate concerns. These approaches do not generally explicitly manage uncertainty, but rather rely on continual re-planning—if the environment does not evolve as expected the planner resets and plans a new sequence of actions. This can lead to failures if environmental dynamics do not evolve as modeled at the task-level.

The approach evaluated in this chapter builds upon the two-level hierarchy of the previous chapter. Following the approach of Chapter 4, we decompose a task into multiple belief spaces structured such that the belief in lower levels informs decision making at higher levels. Each belief space planner is defined by \( S \), the set of world states; \( T \), the conditional transition probability between states; \( O \), an observation function; \( A \), the set of available actions; \( b_k \), the belief distribution over states \( \in S \) at time \( k \); \( r(b_k, A) \rightarrow \mathbb{R} \) a reward function parameterized by the belief distribution and actions; and \( Z \), a state abstraction function \( Z(b_i) \rightarrow z \), where \( b_i \) is the belief of the preceding level. In this chapter we introduce a hierarchy of three levels to control uncertainty in the environment, over object identities, and task completion.
We accomplish this through an *environmental level*, an *intra-object level*, and an *inter-object level*. Each level evaluates uncertainty present in its current state with information theoretic measures. At runtime actions prescribed by each level are evaluated by an *ARBITER* module that selects the action to be physically executed. A diagram of the system is shown in Figure 5.1. These components are described in the following sections.

![Diagram of the system](Figure 5.1: Data flow in a three-level HABP architecture. Belief stabilizes as it progresses up the hierarchy. Each planner selects actions $a^*$ to execute that maximize their respective metrics. An arbitration module selects an action from the set of best actions to execute on the robot.)

### 5.1 Environmental Level

The lowest-level of the three-level hierarchy—the environmental level—maintains belief over candidate object locations while exploring the environment. A candidate object location is a volume of occupied space detected by the robot that is not part of
the known environment. These candidate objects are tracked by maintaining belief in an occupancy map or voxel grid. Belief \( b_{\text{ENV}} \) of occupancy is maintained for cells in the map, where \( b(\text{cell}) = 1 \) when a cell probability of occupancy is certain and \( b(\text{cell}) = 0 \) when the cell is probability of a cell is occupied is 0. \( b(\text{cell}) = 0 \) indicates that the cell is free-space and, therefore, navigable. This yields an underlying state space \( S_{\text{ENV}} \) of \( 2n \), where \( n \) is the number of cells or voxels in the map. Belief in occupancy is updated following the update equations in Section 2.1. The transition function \( T_{\text{ENV}} \) defines how likely cells will change from occupied to freespace and vice-versa. This characteristic is dependent on the properties of the environment. The observation function \( O_{\text{ENV}} \) describes the reliability of sensor feedback and can account for noise due to the distance away from the robot sensed volumes lie. Observations \( z_{\text{ENV}} \) are provided from a perceptual front-end that extracts volumes above the ground plane from point-cloud data [33]. Volumes that are predicted from a given map can be segmented and ignored.

At this level, the robot selects actions (trajectories) to execute from a set of valid trajectories \( A_{\text{ENV}} \). Trajectories can be generated for both the base (mobility) and end-effectors (manipulation) to gain information through visual and tactile observations respectfully. The environmental level implemented in this chapter only considers trajectories that re-positions the mobile base of the robot. These trajectories re-position the robot’s sensory apparatus in the environment in a manner that will investigate the space efficiently. Such search problems have been studied using ideas similar to the ABP [42, 76]. To select an information gathering trajectory to execute these techniques often follow one of two approaches: generate a set of feasible, collision-free trajectories and then execute the best trajectory according to a specified metric (such as entropy or information gain) or optimize a selected trajectory according to a specified metric. Rather than follow these approaches, we will leverage a motion planning technique, the harmonic function path planner [15, 16],
specially formulated to find collision-free trajectories while simultaneously reducing uncertainty by optimizing $r_{ENV}$; information gain.

5.1.1 Harmonic Function Planning and Information Gain

Consider exploration in a partially observable environment. In this setting, a robot is tasked to uncover the contents of the environment while simultaneously avoiding collisions with obstacles. This is typically addressed through frontier-based exploration [81], which seeks discover new information by uncovering the known frontier. We will leverage similar ideas by maintaining a “frontier” of unknown space, which can be non-contiguous. The optimal trajectory for a robot to follow in this problem is one that minimizes collisions with the environment while gaining the most information about the environment. Frontier-based Harmonic function path planning can be applied to address both of these problems concurrently. Harmonic function path planning has several favorable qualities, such as: no local minima, smooth trajectories, completeness, and efficient computation [15, 16]. Trajectories generated from this technique drive the robot toward a goal-set (taking distance to goals into consideration) while minimizing the probability of collisions with obstacles in the environment [14].

These properties can be used to generate collision-free information gathering trajectories. By maintaining a goal set of uncertain cells (cells such that $Pr(\text{cell} = \text{occupied}) \approx 50\%$) as the frontier, the harmonic function approach selects collision-free paths that maximally sweeps the robot’s sensors through the goal set while simultaneously considering the cost (distance to goals) of executing a particular trajectory. Formulated this way, harmonic function path planning selects safe, collision-free trajectories that maximize information gain in a computationally efficient way. These trajectories maximize information gain as generated trajectories will drive the system
towards the largest unknown area. Observing this area would reveal the contents of the space and provide the largest gain in information.

**Theorem 5.1.1.** When unknown regions of the state-space are selected as goal-sets, harmonic function planning techniques select trajectories that maximize information gain while simultaneously maximally avoiding collisions with the environment.

*Proof.* Assume that a trajectory $a$ selected by the harmonic function planner does not maximally acquire information while avoiding obstacles. This implies that the trajectory chosen will not drive the robot to the largest accessible goal-set, which would expose the most new information to the robot. Using the hitting probability result of the harmonic function path planner [14], generated trajectories will maximally avoid obstacles while following the gradient to the goal-set. This gradient will drive the system to the largest accessible goal-set, which will expose the most new information to the robot. Therefore, the trajectory selected by the harmonic function path planner will provide the largest information gain while providing guarantees on collision avoidance.

An example of the harmonic function frontier path planner in a 2D world is shown in Figure 5.2. The robot is represented by a red circle. Initially the robot is unaware of the contents of the environment and must reveal them through search. Obstacles are shown in blue, while unknown regions are green. The value of the harmonic function is represented with the gradient from black to white. White regions are areas that contain the most information gain. The gradient the robot is following (and its relative strength) is shown as a red line drawn from the robot.

### 5.2 Intra-object Level

The intra-object level of the three-level hierarchy manages belief over identities of objects in the environment with regards to a provided model set of objects. Objects
Figure 5.2: Snapshots showing an example trajectory roll-out of a 2D robot (in red) exploring a room using the Frontier-based harmonic function path planner. The trajectory gains information while simultaneously maximally avoiding obstacles (in blue) when unknown areas (in green) are used as goals. The trajectories direct the robot towards the largest unknown areas remaining during search.
are modeled as Aspect Transition Graphs (ATGs), a multi-graph that describes how actions create changes between constellations of observable features (aspects) of an object [48, 66]. Nodes in the graph are aspects, while edges encode the transition probabilities $T_{INTRA}$ between these aspects. Using nodes as state $S_{INTRA}$, belief can be maintained over object identity of ‘hypotheses’—spatially constrained volumes of $\mathbb{R}^3$. Hypotheses are instantiated through a state abstraction function $Z_{INTRA}$, which samples belief of occupied cells. When cells in view are believed to be occupied, a generalized Hough transform [5] computes observation probabilities to known aspects in the model set based upon extracted features present in the volume used as $O_{INTRA}$.

The state abstraction function also captures constraints on observability in the intra-object level based on environmental contexts. If object features were expected to be observed but were not, two possible explanations exist:

1. Object features expected to be reserved are occluded
2. The object we are investigating no longer exists in this volume—i.e. it has been moved

In the first case—object features are occluded—the robot is unable to observe expected features due to reachability or line-of-sight constraints. When an object is occluded, the environmental belief in occupancy is still high, as the agent still believes the corresponding hypothesis volume is occupied. In this case, belief updates do not erode existing distributions but are maintained from the previous time-step. However, if there is high confidence that the volumetric region previously occupied by the hypothesis is now vacant, updates are applied to erode the respective intra-object belief distributions. If intra-object level belief vanishes—that is it becomes zero across the entire state space—the corresponding hypothesis is removed.

Actions in $A_{INTRA}$ change the robot’s relative sensorimotor geometry with regards to a particular object. To control uncertainty amongst the object models, a reward
function $r_{INTRA}$ is specified which maximizes the reduction of future uncertainty in the belief distributions. This is achieved through maximizing the information gain of the next action

$$r_{INTRA} = H(s_k) - \mathbb{E}_{s_{k+1}}[H(s_{k+1}|a_k)]$$ (5.1)

In this way, the robot actively controls the uncertainty over object identity. The implemented intra-object level used a model-set of 19 ARcube ATGs in its state space, yielding a state space of 912 aspect-nodes for each hypothesis. The intra-object level selected actions from these ATGs as described in Section 4.3.1.

### 5.3 Inter-object Level

The top level of the hierarchy manages uncertainty in the spatial precision of the task with regards to object location. The state abstraction function $Z_{INTER}$ samples the maximum likelihood state of the intra-object level distribution to “observe” the expected positions of objects that are relevant to the task at hand. The belief state $b_{INTRA}$ is updated using $O_{INTER}$ and $T_{INTER}$, which are empirically obtained sensor and performance models. Given current belief, this level recommends pick and place actions $\in A_{INTER}$ that rearrange objects in the environment relative to each other such that they maximize the reward function $r_{INTER}$. The reward function used in this work is the negative Kullback-Leibler divergence $D_{KL}$ [50] measured between the current belief of the inter-object level ($b_2$) and a task dependent goal distribution $G$, which is specified a priori. Actions from $A$ which maximize this measure (which minimizes the divergence) are selected for execution.

$G$ specifies the goal positions of targets and the amount of acceptable uncertainty at goals. In this work, the amount of acceptable uncertainty is chosen using empirical models of the robot’s performance $\mathcal{N}(\mu, \Sigma)$ which are captured by this level’s transition and observation functions. The robot can be biased towards faster solutions (at the cost of precision) by penalized by action cost $\lambda$, yielding,
\[
    r_{INTER} = -D_{KL}(b_{inter} \parallel G) - \sum_{i}^{k+l} \lambda_i. \tag{5.2}
\]

When the robot is no longer confident that it can improve the current reward, that is,

\[
r_{INTER_k} \geq r_{INTER_{k+1}} \tag{5.3}
\]

the robot can select a no-operation action \( NOP \) to indicate that it has satisfied task objectives to the best of its ability.

5.4 **Arbiter**

To determine control authority in the three-level hierarchy we will use the same confidence-based subsumption arbitration mechanism introduced in Chapter 4, taking a conservative approach to decision making in the hierarchy by condensing belief bottom-up as required by the task. As such, the robot will explore the environment until it has belief that candidate objects exist. When belief of candidate objects is sufficiently high, the intra-object level will manage uncertainty around object identity to address task requirements. If belief vanishes, control authority will be yielded back to the environmental level. If confidence in required task attributes is met, then the inter-object level will select actions to reduce uncertainty with regards to the overall task by re-positioning objects in the environment. If confidence in the upper-levels is lost, control authority will be yielded to the lowest level with sufficient confidence.

5.5 **Vanishing Belief**

Planning in belief space requires that models capture possible action outcomes and predict future observations in order to maintain the distributions throughout task execution. If models poorly capture the underlying belief dynamics then belief can *vanish*, that is, the distribution of belief over the domain goes to zero. This can
be informative—it might indicate that the robot has assumed an incorrect domain, that models are incomplete, that previous interactions have yielded hallucinations, or that the robot is interacting with a novel object yet to be modeled. Unfortunately, it is difficult to differentiate these cases. Working under the assumption that we are in fact operating with an object from the domain that may be poorly modeled, or that hallucinations have resulted in skewed belief, how can we recover the system?

To better understand candidate recovery techniques, it is informative to understand how such events can occur. Belief distributions are maintained through recursive Bayesian updates. Commonly, this is broken down into two steps, a control update and a measurement update (see Section 2.1). Suppose that at time \( t \), a robot has a prior belief distribution \( b(s_t) \). Using the control update, the robot may predict how any action \( a_t \) available to it at this time will impact the current state. This new predicted belief state will account for all plausible outcomes of the action taken, therefore, it will disperse the previous belief amongst these outcomes. To recover the true belief state, \( b(s_{t+1}) \) from equations 2.1 and 2.2 needs to be corrected with a new observation using an observation model. There are three possible explanations for why a belief distribution would vanish after updating.

1. **Malformed prior**: If a prior \( b(s_t) \) is not plausible it may not predict future states correctly. This mismatch could cause \( b(s_{t+1}) \) to vanish or may cause the observation to completely refute expected outcomes.

2. **Transition model not accurate**: If the transition model \( \text{Pr}(s_{t+1}|a_t, s_t) \) used in the control update does not capture the underlying dynamics it may cause belief to vanish as the resulting observation may refute the expected state.

3. **Observation model is too restrictive**: If the observation model \( \text{Pr}(z_{t+1}|s_{t+1}) \) used in the measurement update does not account for all possible observable outcomes then belief will vanish after the observation update.
These three explanations are the result of poorly modeled belief dynamics. Belief dynamics could have been intentionally (to limit reasoning to more likely states, ignoring edge cases) or unintentionally poorly modeled.

### 5.5.1 Managing Vanished Belief Distributions

A naive way to recover a belief distribution that has vanished is to reset belief. Belief can be reset to an initial prior (e.g., uniform distribution over the domain) or to the distribution just prior to when belief vanished. There are drawbacks to these simple methods of recovery. Resetting to an initial prior is effectively “giving up” and trying again. All information acquired up to the vanish is lost. Resetting belief to the prior maintains the information acquired up to the vanish, but does not work well in situations where the object state has changed between the last two observations—essentially the prior may be malformed and will not be able to explain new observations.

In this work, when belief distributions vanish the system will purge the corresponding abstractions from memory and acquire new abstractions from current sensory inputs—essentially this resets to an initial prior. Any higher-level belief formed from these abstractions will also be purged. This is a reasonable approach in environments that include actions by unmodeled external agents. The actions of external agents are not captured by the belief dynamics of the system, so it is possible that object state will change between observations due to external agent interactions.

### 5.6 Evaluating Long-Term Autonomy under Uncertainty

To test our system’s capability for long term autonomy, we propose a set of mobile manipulation scenarios in unstructured environments where the robot must execute tasks autonomously. These scenarios will be conducted using the uBot mobile manipulator (see Section 2.7) in unstructured laboratory environments in a “tidy-up” task.
This “tidy-up” scenario is analogous to tasks like clearing tables that is suited to our platform. The scenario tests the ability of our system in a simple, but challenging dynamic domain.

In the scenario, the robot must patrol a room, identify objects that are out of place, and return them to a pre-specified position. The robot is placed in a room with an unknown number of objects drawn from a model-set. Initially the poses and identities of objects in the environment are unknown. The robot is provided with the identity of a target object to be put-away. The robot must find candidate objects in the environment, identify them, and put correct ones away at a specified position in the proper orientation. During execution the object may be re-oriented or re-placed at will by an unmodeled external agent. The robot must adapt to these changes and avoid states that could lead to failures. The scenarios will end after roughly four hours of robot deployment. Deployments will pause when the robot drains battery power to unsafe levels. After the batteries have been sufficiently recharged execution will be resumed.

To measure performance in these domains we track total run-time, the number of actions executed, the number of objects correctly stowed (tidied), and the number external interventions that were required during execution. Ideally, we would like to see long run-times paired with high numbers of executed actions and objects stowed while minimizing required external resets due to decision making failures. A key measure of sufficient performance in this domain is the number of autonomous actions executed between resets caused by decision making failures. To the best of our knowledge, state of the art systems can only achieve on the order of tens of actions before human interventions are required. Excepting pauses necessary for re-charging the battery, we will strive for hundreds of actions between resets due to such decision making failures.
5.7 Implementation Details

The evaluation environment consists of a 4.0 m by 6.0 m empty room. The target object to be put-away is an ARcube with a ‘2-1’ aspect—i.e. a ‘2’ feature on the front of the cube and a ‘1’ feature on top of the cube. The goal position in world frame of object features when stowed was $(4.09, 1.58, 0.14)$ m for the front tag and $(4.24, 1.58, 0.28)$ m for the top tag. Specifying the target object in terms of the aspect and not unique identity allows the robot to reason about goals in terms of multiple objects that satisfy particular qualities. This is analogous to the task of putting away all cups in the cupboard, regardless of whether it is a coffee cup or a pint glass, it is considered put-away when placed in the cupboard. An additional distractor object which did not have the attributes required by the task was occasionally introduced into the scene in an attempt to confuse the robot. When each deployment begins the robot localizes to known landmarks in the room. During deployments the robot re-localizes to these landmarks before and after putting objects away.

5.8 Deployment Results

The robot was deployed in the evaluation environment for over four hours. These deployments were interrupted when the batteries dropped to unsafe levels or when external interventions required a reset of the system. The deployments are summarized in terms of length, number of actions executed, items stowed, and the number of failures (external interventions) in Table 5.1. The robot successfully put the target object away and never “put-away” a non-target/distractor object during the deployments. It should be noted that at multiple times during the deployments the object the robot was interacting with was moved or repositioned. As such, the robot was able to put-away a much smaller number of objects than is practically feasible to within the deployment time-frames. The low number of stowed items should not be
Deployment | Duration (h : m : s) | Actions | Items Stowed | Failures
--- | --- | --- | --- | ---
1 | 1:18:29 | 37 | 2 | 2
2 | 0:41:54 | 28 | 1 | 2
3 | 0:16:40 | 12 | 1 | 2
4 | 0:02:20 | 2 | 0 | 1
5 | 0:32:23 | 19 | 3 | 0
6 | 0:45:47 | 29 | 3 | 1
7 | 0:48:05 | 27 | 2 | 1
Total | 4:23:28 | 154 | 11 | 9

Table 5.1: The duration, number of actions executed by the robot, items correctly stowed, and the number of external interventions that were required (failures) across 7 deployments.

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Environmental</th>
<th>Intra-object</th>
<th>Inter-object</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>32</td>
<td>86</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 5.2: Breakdown of the 154 actions executed over seven deployments by hierarchy level.

noted as a shortcoming as it is the direct result of intentional injections of uncertainty into the environment by the experimenter.

Deployment durations varied significantly across trials. Deployments 1 and 5 were ended due to low battery levels. The remaining deployments ended due to necessary external resets. Of note is deployment 4, which ended after interacting with the first object in the scene. 154 actions were executed over all deployments, during which 9 external interventions were required. This yields a failure rate of 5.8%. The number of actions executed by each level during deployments is shown in Table 5.2.

Action counts indicate how many times control authority was obtained by a particular level. Although from Table 5.2 it may appear that the robot did not search
the space extensively, these counts only indicate when the robot began to search for objects and does not represent the duration of the search. Higher activations of the environmental level indicated that the object(s) the robot was interacting with was removed or relocated by experimenter interactions. In deployment 1, a bug in the implemented localization code induced enough uncertainty in the inter-object level that the robot repeatedly attempted to re-place the target object in the correct position. Although this did not adversely affect the robot—it believed that the object was misplaced and attempted to mitigate the uncertainty present in its internal state—it could be viewed as sub-optimal behavior. This bug was fixed for later deployments.

5.8.1 Interventions

The robot encountered nine failures which required interventions during the deployments. These failures are shown in Figure 5.3. Figure 5.3a shows a required intervention due to hardware failure (a faulty connector in the robot’s force-torque sensors). Figure 5.3b shows a failure where the robot attempted to manipulate a previously moved object. This failure is attributable to the HABP system and will be discussed in more detail in the following section.

Most of the remaining failures, shown in Figures 5.3c-h, occurred while manipulating objects in the environment as prescribed by the intra-object level. According to the belief state at the time of these failures, these were legible actions. However, due to uncertainties in the underlying controllers these actions failed during execution. Similar to Ku et al. [49], if the object-models used for planning were richer (contained more fine-grained transitions) the intra-object level could have detected the inappropriate evolution of state and avoided the failures. The external reset shown in 5.3(i) occurred due to an error in interprocess communication.
Figure 5.3: Images showing the failures experienced during deployments. Only the failure shown in (b) is attributable to the HABP system. A bug in the implementation prevented messages from being transferred between planning processes. This was rectified in subsequent deployments.
5.8.2 Experimenter Interactions

During the seven deployments the experimenter interacted with experimental objects a total of 37 times. These interactions re-positioned and re-oriented objects in the environment, and/or inserted or removed distractor or target objects. The number of interactions per deployment are documented in Table 5.3.

The one failure attributable to direct interaction (Figure 5.3b) occurred in the first deployment. The cause of the failure was an erroneous implementation of the state abstraction function between the environmental and intra-object levels. Snapshots of this failure and the events leading up to it are shown in Figure 5.4. While searching the environment, the robot observed an object (a). While the robot traversed towards the object to interact with it (b-g) an experimenter pushed the object to a new position outside the robot’s observable field of view (b-e). Due to an implementation error, although the environmental level belief provided little support for hypothesis existence, the intra-object level belief was not correctly updated. This caused the robot to grasp empty space (g), which required intervention to correct. This im-

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Number of interactions</th>
<th>Failures due to interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>12*</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>37</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.3: The number of interactions taken by the experimenter to alter the current environmental state during execution. These interactions moved, withdrew, or added objects in the scene. Despite these attempts to foil the robot, it was able to manage the induced uncertainty with the three-level hierarchy to still address the task. *In deployment 7, three experimenter interactions did not affect the robot at all, as two interactions inserted and withdrew an unobserved object. The third caused a disturbance that was not observed due to an un-related failure that ended the trial prematurely.
plementation error was fixed in subsequent deployments. This failure highlights the impact that a correctly implemented HABP has in such situations—the robot avoids executing actions akin to those shown in Figure 1.1.

Two of the 37 interactions that avoided decision making failures by reasoning in the hierarchical framework are shown in Figures 5.5 and 5.6. In both cases, after the object was removed by an experimenter the environmental level belief updated and provided little support that the volume expected to be occupied by the hypothesis was still occupied. This belief was then sampled by the intra-object level through the state abstraction function to correctly purge the hypothesis from memory. Control authority was then yielded back to the environmental level to search for more objects.
Figure 5.4: The sole failure after experimenter interaction (b-e) during the first deployment. Due to an implementation error, although the environmental level detected that the object had been removed the intra-object level belief was not correctly updated, causing the robot to grasp empty space (g), which required intervention to correct.
Figure 5.5: Example behavior after experimenter interaction. After observing the object in (a), the robot executes an action to observe the object from another vantage point (b-e). An experimenter removes the object in (c). By sampling the environmental level’s belief of occupancy (which provided little support for the expected volume), the intra-object level determined the object had been removed (e). Execution was then yielded to the environmental level to find new objects (f).
Figure 5.6: Example robust behavior after experimenter interaction due to reasoning in the hierarchy. While manipulating an object (a-d) an experimenter removes the object (c-d). By sampling the environmental level’s belief of occupancy, the intra-object level determined that the object had been removed, as no support for a volume existed in the expected position (e). The intra-object level execution was then yielded to the environmental level, which began executing trajectories to find new objects to interact with (f).

5.9 Discussion

The three-level hierarchy demonstrated in these deployments commits to control decisions only when belief in the various levels of abstraction have stabilized. By
selecting which level of the hierarchy possesses control authority through a confidence-based subsumption arbitration, the system is able to manage uncertainty across the levels of the hierarchy. As a result, robots using the three-level HABP planning architecture will only interact with objects that have strong environmental evidence for existence and only use objects in task roles that satisfy task requirements.

Despite repeated attempts by experimenters to frustrate the robot, it completed the specified ‘tidy-task’ over deployments lasting hours during which over 150 actions were executed. During these deployments only nine failures requiring external interventions occurred. Two of the interventions were due to system-level failures such as cabling or inter-process communication. Only one failure due to improper decision making occurred, which was attributable to a software bug. The remaining six failures occurred due to uncertainty in lower level controllers. If the models used in planning contained finer-grained transition dynamics such failures could be avoidable. As such, the failures observed during LTA deployments do not provide negative evidence that the architecture is able to cope with uncertainties present in open worlds. These outcomes provide strong support that decision making using the HABP framework can enable robust autonomy for robots acting in dynamic, partially observable environments.
CHAPTER 6

CONCLUSION

So long as robots continue to ignore the impact of uncertainty and risk during execution, autonomous robots that are capable and reliable in unstructured environments will continue to only occupy our imaginations. In order to successfully navigate real-world, unstructured environments robots must tend to uncertainty as it impacts task performance. The techniques described in this dissertation can be used to improve the performance of autonomous systems that face uncertainty from the partially observable, unstructured, stochastic environments of the real world. To achieve this, this dissertation has presented three core contributions to autonomous decision making for robots that interact with partially observable, unstructured environments:

1. A hybrid planning framework to accomplish tasks using symbolic planning that leverages robust interaction via belief space planning methods.

Chapter 3 presented a hybrid planning system that uses the ABP to manage uncertainty within objects to enable more robust symbolic plans in mobile manipulation tasks. However, as described, prematurely committing to symbolic representations lead to brittle, error-prone behavior when uncertainty compromises the presumed semantics of the world. To address this,

2. A recursive definition of model-based belief space planners that provides a basis for hierarchical organization and supports multiple levels of abstraction.
A hierarchical definition of model-based belief space planners was presented in Chapter 4 that manages uncertainty across different levels of abstraction. Through the introduction of the state abstraction function $Z$, higher levels of a hierarchy can form observations predicated on the belief of lower levels. This allows the system to be sensitive to attributes of the different spaces as required by a task. The introduction of the confidence-based subsumption arbiter allows the system to be sensitive to uncertainty as it puts the overall task at risk by yielding control authority to the lowest level of the hierarchy where minimum confidence has been established. This framework was demonstrated in

3. Example hierarchies that manage uncertainty in autonomous behavior that operates reliably for an extended period of time (on the order of hours) without external intervention.

The example two-level (Chapter 4) and three-level (Chapter 5) hierarchies presented methods for autonomous execution of robot systems in unstructured, partially observable environments. The hierarchical belief space planning architecture introduced and evaluated shows great promise in supporting robust, reliable autonomous systems. The example hierarchies illustrate how this architecture leads to robust run-time behaviors over long-term deployments in mobile manipulation domains which surpass state-of-the-art systems. Although the robot did encounter failures that required resets during these deployments, the errors were not attributable to ill-founded controls decisions. This dissertation is a promising direction to address the issues prohibiting robots—in particular mobile manipulators—from being successful “in the wild.”

6.1 Recommendations for Future Work

The work performed over the course of this dissertation has exposed several new areas for future research and investigation. These topics cover a wide spectrum of research areas, from reinforcement learning to human robot interaction. They also
range from applying the techniques presented in this dissertation in unique ways to shoring up the underlying representations that the belief space planners rely on.

6.1.1 Autonomous Model Building

Model based belief space planners like the ABP depend on internal models to update belief and predict the impact of future actions on tasks. The models used in this dissertation were largely hand-built using empirically obtained values. Autonomously constructing and learning these models was not investigated. As robot action sets become larger and the objects and tasks we wish the robot to interact with and complete become more complex, it will become increasingly important to automate model building. A promising direction investigated previously by Wong and Grupen [79] to self-supervise model learning offline using intrinsically motivated model-based reinforcement learning could be leveraged to automate model acquisition. A challenging extension of that work would be to adapt and/or extend the model-set online based on current interactions while addressing tasks.

Many of the errors encountered during deployments in Chapter 5 can be attributed to a lack of fine-grained transition dynamics in the ATGs used by the planner during deployments. Ku et al. modeled fine-grained transition dynamics that could avoid error states during execution [49]. However, identifying and learning appropriate fine-grained transition dynamics is non-trivial. In planning domains with model-sets containing highly similar objects (such as ARcubes), these fine-grained transitions may not expose new information. As informative state would now lie several ply away from the current state, multi-ply planning is required. This impacts planning negatively, as a combinatoric explosion of future states needs to be considered. However, this may not be the case in model-sets with dissimilar objects, as fine-grained transition dynamics would likely rapidly expose new information.
6.1.2 Arbitration Mechanisms

In the implemented hierarchies in this dissertation, control authority was determined at run-time using a confidence-based subsumption approach. This enabled the robot to manage uncertainty across the different levels of abstraction in the hierarchy to levels defined by task requirements in a conservative manner. However, other metrics could determine control authority at run-time. An ideal metric would capture the impact of executing actions prescribed by a particular level on the confidence of completing the overall task. In a HABP implementation with an arbitrary number of levels, such a metric could have the form:

\[ \frac{\partial r_n}{\partial r_i} = \frac{\partial r_n}{\partial r_{n-1}} \frac{\partial r_{n-1}}{\partial r_{n-2}} \cdots \frac{\partial r_{i+1}}{\partial r_i} \]

where \( \frac{\partial r_n}{\partial r_i} \) captures the impact changes in reward at level \( i \) have on the overall task captured by level \( n \). Such a relationship is difficult to capture analytically. However, it may be possible to learn such relationships using reinforcement learning techniques. Learned policies could select which level of the hierarchy should be executed dependent on the belief over all levels in the hierarchy.

6.1.3 Explainable AI

As autonomous systems become embedded in society it will be increasingly important to effectively communicate intent to users and to explain previous decisions. This component of human-robot interaction research has been receiving increasing attention [23]. Belief-based frameworks such as HABP could provide the basis for an intuitive, explainable AI system. Using such an approach, a robot could explain why it made previous decisions based upon past and current belief states of the hierarchy. By unrolling forward plies of the search tree the robot could communicate the intent of future actions. In this sense, HABP could be an informative tool to understand why an autonomous agent has, is, or will select particular actions to be executed.
In situations where guaranteed performance is required Explainable AI can also be leveraged to request additional guidance from the human in the spirit of Tellex et al. [74]. This is crucial in situations where the robot is unable to verify its current state within some arbitrary threshold. In these cases, a possible strategy could be for the robot to abort execution to a safe state and wait for human input or re-schedule other tasks similar to Chien et al. [12]. Following this approach could limit the fidelity required in predictive models but would create a dependence on preventative interventions from humans. As such, in deployed systems it should be used in conjunction with high fidelity models to increase autonomous viability as much as possible.

6.1.4 Recovering Vanished Belief Distributions

The implemented hierarchies in this dissertation relied on a conservative, pragmatic approach to vanishing belief distributions—when belief vanished the corresponding representations throughout the hierarchy would be purged (as described in Section 5.5.1). This naive approach could be improved upon to leverage environmental contexts. Rather than purge the corresponding abstractions completely, the system may be able to recover the belief if future observations are matched to expectations given the higher-level models. This could be beneficial in situations where the object investigated by the robot was moved between interactions. Recovering the belief if the object is rediscovered could save planning and execution time by leveraging previously obtained information. Unfortunately, such matching or correspondence problems are non-trivial in practice.

6.1.5 Temporal Planning in Belief Space

In this dissertation the temporal implications of actions were largely ignored. Additionally, in experiments only one high-level task was specified to the robot during each deployment. In reality, for robots to be deployed effectively they will need to
manage multiple competing high-level tasks at runtime. Agents will need to be able to effectively estimate how long tasks will take to be completed as well as respect temporal constraints on execution (e.g. task A can only be completed in the morning). Extending the hierarchical belief space architecture proposed in this work to handle these situations could be fruitful. A possible implementation could extend the hierarchy of Chapter 5 to include a temporal level above the environmental level. Such an extension would require new models that capture spatio-temporal relationships of objects used in tasks. Alternatively, parallel hierarchies could be switched between to accomplish tasks within temporal constraints. In this case, arbitration mechanisms could capture temporal dependencies and activate the necessary hierarchy when required.

6.1.6 Additional Extensions to the Three-Level Hierarchy

The implemented hierarchy of Chapter 5 had three levels of planners that managed uncertainty at environmental, intra-object, and inter-object levels. This hierarchy could be extended in both directions in future work, adding levels above or below the defined hierarchy. These added levels would allow the robot to address more complex tasks and enable more seamless execution. To enable more complex tasks that involve sub-assemblies, an additional ABP could be defined above the inter-object level to manage these sub-assemblies that are created by the inter-object level.

Below the defined three-level hierarchy, an additional ABP could be defined to manage the robot’s pose uncertainty through localization. Actions could be trajectories to generate targeted, informed observations that help localize the robot as explored by Stachniss et al. [72] and Roy [61]. Formulating localization in the HABP architecture can capture the dependencies between required pose uncertainty and action preconditions. Using a similar confidence-based subsumption arbiter defined in this dissertation, the robot would first obtain low entropy estimates of pose before
allowing higher-levels to execute and would re-acquire control authority when the robot looses confidence of its pose.
BIBLIOGRAPHY


