Foraging, Prospecting, and Falsification -
Improving Three Aspects of Autonomous Science

P. Michael Furlong

21 March 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr. David Wettergreen, Chair
Dr. Jeff Schneider
Dr. Nathan Michael
Dr. David Thompson (JPL)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2018 P. Michael Furlong

This work was supported by NSERC, And the following Projects: Robotic Investigation of Subsurface Life in the Atacama Desert (NNX11AJ87G), Reliable Autonomous Surface Mobility (NNX10CF52P), The Mojave Volatiles Prospector, Resource Prospector, ARADS, ICICLES

Opinions expressed in this document are solely those of the author.
Abstract

Robots exploring the subsurface ocean of Europa, for example, will not have reliable communications with scientists on Earth. Robots exploring with unreliable communications must conduct scientific exploration autonomously. Automation of scientific exploration requires both opportunistic and deliberative decision-making algorithms.

Opportunistic decision making chooses to investigate events or phenomena which could not have been anticipated from prior knowledge but which may yield valuable scientific data. Deliberative decision making uses prior knowledge to plan observations that increase information. Approaches to deliberative and opportunistic science autonomy that work in the laboratory may not work in the field.

This thesis presents three algorithms designed to improve the performance of robots conducting autonomous science investigations. The first algorithm, foraging, improves opportunistic responses by deciding between sampling immediately available objects or searching for better options. Foraging moves science autonomy beyond simply responding to matched templates or anomalous data. The work recognizes that robots may not get to choose which objects they can to sample, but must deal with what they encounter. Our approach has increased performance in selecting objects to sample when sampling costs are high, without neglecting opportunities when sampling costs are low.

The second algorithm addresses how to effectively conduct prospecting without relying on either arbitrary thresholds or responding to anomalies. Threshold-based algorithms cannot distinguish between anomalies and true changes in the distribution driving sensor readings. We present an algorithm that directly poses the question of whether or not a change has occurred. The change detection algorithm developed in this thesis encodes a level of confidence that a change has occurred, based on data collected by the robot. This can improve the efficiency of the investigation.

The third algorithm represents a new approach to information gathering based on falsification. Recognizing scientists come to missions with hypotheses formulated, the algorithm uses those hypotheses to choose sampling actions that help determine which hypothesis is most credible. Prior information gathering approaches consider one or fewer hypotheses, and focus mainly on sampling the hypotheses’ domain. We show that our approach biases the belief in hypotheses towards the most credible one.

The thesis proven in this work is that by accounting for operational conditions, science autonomy algorithms will be improved. Each of these algorithms improve components of autonomous science, and thereby the process as a whole.
Acknowledgments
## Contents

### 1 Introduction

1.1 Why Automate Science? ......................................................... 4
1.2 Thesis Statement ................................................................. 5
    1.2.1 Opportunistic Sampling of Discrete Objects (Foraging) - Chapter 3 .... 6
    1.2.2 Opportunistic Sampling In a Scalar Field (Prospecting) - Chapter 4 .... 7
    1.2.3 Experiment Design for Hypothesis Falsification (Informative Experiment Planning) - Chapter 5 ... 8
1.3 Scope of Work .................................................................... 8
1.4 Summary ............................................................................ 9

### 2 Related Work

2.1 Passive Sampling ................................................................. 11
2.2 Opportunistic Sampling ....................................................... 13
    2.2.1 Opportunistic Sampling of Discrete Objects (Foraging) ................... 14
    2.2.2 Opportunistic Science in Fields - Prospecting ............................... 19
2.3 Informative Path Planning ..................................................... 21
2.4 Hypothesis Generation ......................................................... 26
2.5 Summary ........................................................................... 30

### 3 Opportunistic Sampling of Discrete Objects (Foraging)

3.1 Prior Work ....................................................................... 34
    3.1.1 The Secretary Problem ....................................................... 34
    3.1.2 Multi-armed bandits .......................................................... 35
    3.1.3 Optimal Foraging .............................................................. 36
    3.1.4 Opportunistic Science ......................................................... 36
3.2 Method ............................................................................ 37
    3.2.1 Algorithms ................................................................... 38
3.3 Experiments ..................................................................... 41
    3.3.1 Experiment 1 - Uniform Arrival Distribution, Different Underlying Distributions ....................................................... 42
    3.3.2 Experiment 2 - Skewed Arrival Distribution with Identical Underlying Distributions ....................................................... 42
    3.3.3 Experiment 3 - Skewed Arrival Distribution with Distractor Object ... 43
### 3.3.4 Experiment 4 - Skewed Arrival Distribution with Random Underlying Distributions

- Page: 44

### 3.3.5 Experiment 5 - Distribution Change

- Page: 44

### 3.4 Results

- Page: 44

#### 3.4.1 Experiment 1 Results - Uniform Arrival Distribution, Different Underlying Distributions

- Page: 45

#### 3.4.2 Experiment 2 Results - Skewed Arrival Distribution, Identical Underlying Distributions

- Page: 46

#### 3.4.3 Experiment 3 Results - Skewed Arrival Distribution with Distractor Object

- Page: 50

#### 3.4.4 Experiment 4 Results - Skewed Arrival Distribution with Random Underlying Distributions

- Page: 53

#### 3.4.5 Experiment 5 Results - Underlying Distribution Change

- Page: 56

### 3.5 Discussion

- Page: 57

### 3.6 Summary

- Page: 59

### 4 Opportunistic Sampling in a Scalar Fields (Prospecting)

- Page: 61

#### 4.1 The Mojave Volatiles Prospector Project

- Page: 62

#### 4.2 Prior Work

- Page: 63

#### 4.3 Method

- Page: 69

##### 4.3.1 Algorithms

- Page: 70

##### 4.3.2 Execution Time Comparison of SPRT to MCMC-Bayesian Change Detection

- Page: 77

##### 4.3.3 Experiments

- Page: 81

##### 4.3.4 Performance Metrics

- Page: 85

#### 4.4 Results

- Page: 85

##### 4.4.1 Experiment 1 Results - Effect of Magnitude of Change in the Underlying Distribution

- Page: 87

##### 4.4.2 Experiment 2 Results - Effect of Delay of Change Onset

- Page: 90

##### 4.4.3 Experiment 3 Results - Real MVP Data

- Page: 93

##### 4.4.4 Experiment 4 - Effect on Performance in 2D Operations

- Page: 104

##### 4.4.5 Discussion

- Page: 109

#### 4.5 Summary

- Page: 110

### 5 Global Planning for Hypothesis Falsification

- Page: 113

#### 5.1 Prior Work

- Page: 116

#### 5.2 Method

- Page: 120

##### 5.2.1 Belief in Hypotheses

- Page: 120

##### 5.2.2 Site Selection Algorithm

- Page: 121

##### 5.2.3 Map Generation

- Page: 125

##### 5.2.4 Experiments

- Page: 125

#### 5.3 Results

- Page: 127

##### 5.3.1 Hypothesis Falsification with Noise-Free Observations

- Page: 127

##### 5.3.2 Hypothesis Falsification with Noisy Observations

- Page: 129

#### 5.4 Deployment in Chile

- Page: 131
# List of Figures

1.1 Science as a Process ............................................... 1  
1.2 Illustration of foraging scenario ................................. 6  
1.3 Illustration of prospecting scenario .............................. 7  
3.1 A lawn mower, or boustrophedon, trajectory .................... 32  
3.2 Experiments 1.1-1.5 - Foraging vs Greedy, uniform arrival distribution ......................................................... 45  
3.3 Experiments 1.1-1.5 - Foraging vs Uniform, uniform arrival distribution ................................................................. 46  
3.4 Experiment 3.1 - 3.5 Foraging vs Greedy, unbalanced arrival distribution ........................................................... 47  
3.5 Experiment 3.1 - 3.5 Foraging vs Uniform, unbalanced arrival distribution ............................................................... 48  
3.6 Experiment 3.6 - 3.9 Foraging vs Greedy, Zipf arrival distribution ................................................................. 49  
3.7 Experiment 3.6 - 3.9 Foraging vs Uniform, Zipf arrival distribution ............................................................... 50  
3.8 Experiment 4.1-4.4 - Foraging vs Greedy, unbalanced arrival distribution ............................................................... 51  
3.9 Experiment 4.1-4.4 - Foraging vs Uniform, unbalanced arrival distribution ............................................................... 51  
3.10 Experiment 4.5-4.8 - Foraging vs Greedy, Zipfian arrival distribution ............................................................... 52  
3.11 Experiment 4.5-4.8 - Foraging vs Uniform ........................................................ ............................................................... 53  
3.12 Experiments 5.1-5.4 - Foraging vs Greedy, unbalanced arrival distribution ............................................................... 54  
3.13 Experiments 5.1-5.4 - Foraging vs Uniform, unbalanced arrival distribution ............................................................... 55  
3.14 Experiments 5.5-5.8 - Foraging vs Greedy, Zipfian arrival distribution ............................................................... 55  
3.15 Experiments 5.4-5.8 - Foraging vs Uniform, Zipfian arrival distribution ............................................................... 56  
3.16 Experiment 3 - Detecting distribution change yields statistically significant performance improvement ......................... 57  
4.1 KReX2 in the Mojave desert ............................................. 63  
4.2 Execution time for SPRT vs MCMC change detection ............... 79  
4.3 Change detection performance for SPRT vs MCMC .................. 80  
4.4 Example water map simulated from MVP data ...................... 84  
4.5 Evolution of estimated time change, $t_{cp}$ vs time ................... 86  
4.6 Experiment 1 - False Negative Rates ................................ 87  
4.7 Experiment 1 - False Positive Rates ................................ 88  
4.8 Experiment 1 - Mean Error Rate ..................................... 89  
4.9 Experiment 2 - False Positive Rate .................................. 90  
4.10 Experiment 2 - False Negative Rate ................................. 91  
4.11 Experiment 2 - Mean Error .......................................... 93  
4.12 Algorithm Performance on MVP Data from 2014-10-17 ............ 94  
4.13 Algorithm Performance on MVP Data from 2014-10-18 ............ 95
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Experiment 1 - Parameters for underlying distributions</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>Experiment 2 - Unbalanced arrival distributions</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Experiment 2 - Zipfian arrival distributions</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Experiment 4 - Parameter Settings</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>The underlying distributions used in experiment 5. For Experiments 5.1 to 5.4</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Improvement in performance from adaptive sampling from [Ferri et al., 2010]</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Analysis of improvement due to adaptive threshold from [Ferri et al., 2010]</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Effect size of SPRT vs MCMC</td>
<td>81</td>
</tr>
<tr>
<td>4.4</td>
<td>Experiment 1 - Algorithm Parameters</td>
<td>82</td>
</tr>
<tr>
<td>4.5</td>
<td>Experiment 2 - Algorithm Parameters</td>
<td>82</td>
</tr>
<tr>
<td>4.6</td>
<td>Experiment 3 - Algorithm Parameters</td>
<td>83</td>
</tr>
<tr>
<td>4.7</td>
<td>Experiment 4 - Algorithm Parameters</td>
<td>83</td>
</tr>
<tr>
<td>4.8</td>
<td>Experiment 3 - Performance Summary</td>
<td>103</td>
</tr>
<tr>
<td>4.9</td>
<td>Experiment 3 - Precision and Recall Scores</td>
<td>103</td>
</tr>
<tr>
<td>4.10</td>
<td>Experiment 3 - Precision and Recall Effect Sizes</td>
<td>104</td>
</tr>
<tr>
<td>4.11</td>
<td>Experiment 4 - AIMs deployed and maxima observed</td>
<td>107</td>
</tr>
<tr>
<td>4.12</td>
<td>Experiment 4 - AIMs deployed and maxima observed effect size</td>
<td>107</td>
</tr>
<tr>
<td>4.13</td>
<td>Experiment 4 - AIM effectiveness</td>
<td>108</td>
</tr>
<tr>
<td>4.14</td>
<td>Experiment 4 - Difference in effectiveness and effect size</td>
<td>108</td>
</tr>
<tr>
<td>5.1</td>
<td>Similarity of hypotheses to true maps</td>
<td>126</td>
</tr>
<tr>
<td>5.2</td>
<td>Hypotheses used in experiments</td>
<td>126</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiment 1 - Results without Sensor Noise</td>
<td>128</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiment 2 - Results without Sensor Noise</td>
<td>129</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiment 3 - Results without Sensor Noise</td>
<td>129</td>
</tr>
<tr>
<td>5.6</td>
<td>Experiment 1 - Results with Sensor Noise</td>
<td>130</td>
</tr>
<tr>
<td>5.7</td>
<td>Experiment 2 - Results with Sensor Noise</td>
<td>131</td>
</tr>
<tr>
<td>5.8</td>
<td>Experiment 3 - Results with Sensor Noise</td>
<td>131</td>
</tr>
<tr>
<td>5.9</td>
<td>Drill Hole Coordinates and Halite Content</td>
<td>133</td>
</tr>
<tr>
<td>C.1</td>
<td>SPRT 2 vs MemThreshold False Negative Rate</td>
<td>161</td>
</tr>
</tbody>
</table>
C.2 SPRT 4 vs MemThreshold False Negative Rate . . . . . . . . . . . . . . . . . . . 162
C.3 SPRT 8 vs MemThreshold False Negative Rate . . . . . . . . . . . . . . . . . . . 162
C.4 SPRT 2 vs MemThreshold False Positive Rate . . . . . . . . . . . . . . . . . . . 163
C.5 SPRT 4 vs MemThreshold False Positive Rate . . . . . . . . . . . . . . . . . . . 163
C.6 SPRT 8 vs MemThreshold False Positive Rate . . . . . . . . . . . . . . . . . . . 164
C.7 SPRT 2 vs MemThreshold Mean Error . . . . . . . . . . . . . . . . . . . . . . . 164
C.8 SPRT 4 vs MemThreshold Mean Error . . . . . . . . . . . . . . . . . . . . . . . 165
C.9 SPRT 8 vs MemThreshold Mean Error . . . . . . . . . . . . . . . . . . . . . . . 165
C.10 The average reduction in false positive rate due to using the SPRT algorithm vs
the Memory Threshold algorithm . . . . . . . . . . . . . . . . . . . . . . . . . 166
C.11 The reduction in FN rate from the Memory Threshold to the SPRT algorithm . . 166
C.12 The effect size of the reduction in mean error as the change point is varied . . . 167
C.13 Experiment 4 - Maxima observed on Maps by Algorithms . . . . . . . . . . . . 168
C.14 Experiment 4 - AIMS deployed on Maps by Algorithms . . . . . . . . . . . . . 169
List of Algorithms

2.1 A skeleton of the hypothesis search algorithm used in Eureqa. hypothesis_generation is their symbolic regression function, and the score function is the Akaike information criterion. ........................................ 28

2.2 A skeleton of the hypothesis search algorithm used in the automatic statistician. The generate_hypothesis function is a Levin search through the space of kernel functions, and the score function is the Bayesian Information Criterion ......................... 29

3.1 Greedy Sampling Strategy ................................................. 39

3.2 Uniform Sampling Strategy ................................................. 39

3.3 Foraging Sampling Strategy ................................................ 40

3.4 Foraging Change Detection ................................................ 41

4.1 Schematic of threshold adaption rule ..................................... 65

4.2 Threshold change detection ................................................ 71

4.3 Memory Threshold change detection ..................................... 72

4.4 Adaptive Threshold change detection ..................................... 73

4.5 Adaptive threshold update algorithm ..................................... 74

4.6 Relative change detection .................................................. 75

4.7 SPRT change detection ..................................................... 77

5.1 Site selection algorithm .................................................... 122

5.2 Spatial mutual information sampling ..................................... 122

5.3 Hypothesis falsification reward function ................................. 124

5.4 Map generation algorithm .................................................. 125
Chapter 1

Introduction

A natural end-goal of the project of science autonomy is a fully autonomous robot that conducts scientific inquiry without human supervision. However, science autonomy is not a single problem to be solved. In this thesis we demonstrate improvements in different aspects of a robot scientist by being aware of the operational conditions encountered in field work. We examine three different aspects of the scientific process in the context of planetary science missions. Before we discuss the techniques used in this research, we present the notion of science as a process, and explain which aspects of that process we are improving.

![Figure 1.1: The cycle of scientific process. Hypotheses imply experiments to be conducted. Experiments produce observations. Observations update or produce new hypotheses.](image-url)
The automation of the scientific process includes many different activities, but at the minimum it must include generating hypotheses from data, designing experiments to falsify those hypotheses, and selecting (and conducting) informative experiments - for a measure of information preferred by the experimenter. One can connect these activities in a cycle, as depicted in Figure 1.1

The activities in the cycle of scientific inquiry act upon three artefacts: hypotheses, experiments, and observations. We define these terms as follows:

**Hypotheses** are models of some phenomena. Hypotheses must be falsifiable – the must make predictions about the world, that are either implicitly or explicitly observable. For example, a model of the spatial distribution of geologic material predicts that at specific locations there will be specific mixtures of materials. In this document the words “hypothesis” and “model” are used interchangeably. To represent it formally, we assume that a hypothesis, \( h \), maps an input domain, \( X \), to a probability distribution over potential observations, \( P(Z) \).

In symbols:
\[
    h : X \rightarrow \{Z \rightarrow [0, 1]\}.
\]

**Experiments** are actions that the scientist can take that result in observations. It is important to ensure that these experiments are informative with respect to the hypotheses being tested. It is also important to ensure that experiments do not reproduce each other unnecessarily. In this document the word “experiment” and “action” are used interchangeably. In symbols we write \( a : X \rightarrow Z \). In this thesis we consider only one kind of experiment, those that collect an observation, \( Z \), at a given point in the domain of the hypothesis, \( X \).

**Observations** are the results of experiments. Observations are the data that are used to construct or update hypotheses. “Observations” and “data” and “samples” are all phrases that will be used interchangeably in this document. In symbols we would write an observation as an input-output pair, \((x, z)\), where \( x \in X, z \in Z \). The \( i^{th} \) observation would be \((x_i, z_i)\). \( x \) and \( z \) may be vector-valued, they also may also be random variables.

There is disagreement on which is these artefacts have primacy in the scientific process. Karl Popper would have us believe that hypotheses are the currency of scientists (Popper, 2005). Ian Hacking suggests that experiments are the most important object (Hacking, 1983). We suggest that hypotheses, experiments, and data are all equally necessary components of scientific theory, without one taking primacy over the others.

Science as a process is an iterative loop that one may join in at any point and that all three are perfectly valid starting points. Robot scientists will need to manipulate these artefacts when operating autonomously. Hence, roboticists will need to identify the processes that the robot scientist use to move between these entities.

The process going from hypotheses to experiments is called **Experiment Design**. For example, mutual information sampling is a type of experiment design. It identifies an action that can be taken at a given point in the input space of a hypothesis. Mutual information sampling (Lindley, 1956) also has the advantage of scoring the different candidate actions so they can be prioritized. Another type of experiment design is random selection (Lind, 1757; Peirce and Jastrow, 1884). Random sampling reduces the effect of biases on the selection process and has the advantage of forcing the experimenter to more thoroughly explore the domain of the hypothesis.

The process going from experiment to observations is called **Experiment Selection**. The distinction between experiment design and experiment selection is this: Experiment design asks "of all the possible actions, which ones are most informative?". Experiment Selection asks of
"Of all the most informative actions, and given some budget, which are the next best K actions to take?". We can sub-classify experiment selection into groups of reactive and deliberative experiment selection. Both deliberative and reactive experiment selections can be made with respect to the constraints of mission resources.

Reactive experiment selection, also called opportunistic science, determines a sampling action once an opportunity to conduct an experiment has been identified. Reactive sampling actions can only be decided in situ as one does not know what opportunities to conduct experiments will be encountered before one explores the environment.

Deliberative experiment selection, determines a set of experiments to be conducted, and then seeks out opportunities to conduct those experiments in the world. Deliberative sampling actions are planned ahead of time in an attempt to maximize the information gained.

The process going from observations to hypotheses is called Hypothesis Generation. Simple structured machine learning – i.e. linear regression – is a form of hypothesis generation. It chooses the hypothesis (weight settings) most likely given the constraints of the data to be explained and the family of hypothesis (linear functions) specified by the experimenter. Choosing the structure of the hypothesis is part of the hypothesis generation and so model selection must also be considered part of the hypothesis generation.

By iteratively engaging hypothesis generation, experiment design, and experiment selection, robots can produce and test hypotheses, thereby improving their understanding of the phenomena they are investigating. Science autonomy, then, must include research into systems that do one or more of experiment design, experiment selection, and hypothesis generation.

In laboratory settings one has to consider the resources that are available to the robot scientist. A limited volume of reagent, for example, may change how one would prioritize different experiments. If a robot scientist is conducting field work there are even further constraints. Limitations on time or power resources affect how far a vehicle can travel in a day, or whether or not it can backtrack to conduct a follow-up investigation. Noisy sensors complicate the interpretation of data. The robot scientists may not be able to choose which experiments it is able to conduct. There are conditions which exist outside of the laboratory which add additional complications to autonomously conducting science.

In this thesis we address two cases of reactive experiment selection/opportunistic science, which we call “foraging” and “prospecting”, in the context of ground vehicle operations. We define foraging to be the execution discrete sampling actions in response to observing discrete objects in the environment in an attempt to learn distributions underlying those objects. We define prospecting to be the execution of discrete actions in response to observations of a field which the robot is moving through. We also address an experiment design approach which is based on the theory of falsification, which we then use to direct global exploration of a lander capable of re-ignition.

Of course, this is not an exhaustive listing of all the processes that go into science, and it does not address how multiple robot scientists would coordinate their behaviour. There are problems of perception, sample collection and manipulation, hypothesis generation, and communication of findings which are not addressed in this thesis.
1.1 Why Automate Science?

Given that we have a framework for describing science as a process, we should also address why one would bother automating science. There are at least three reasons to automate the scientific process. First, because future flight missions will require it. Second, because human agents demand it. Third, because it poses a truly unique computational problem. The first two reasons are pragmatic, while the third is more interesting from a philosophical standpoint.

Future Missions Require Science Autonomy

NASA, and other space faring science agencies, will require autonomous science simply because the strain on productivity that communications latency places on missions. The further afield a robot explorer travels the lower bandwidth it will have available to communicate back to Earth, and the greater the latency will become. Including humans in the decision making loop will require either designing robots to survive the decision making cycle or forcing scientists to make less nuanced decisions, potentially leading to less effective mission. Further, with profound delays in communications the likelihood of serendipitous discovery will drop precipitously.

A reliable autonomous scientist reduces science missions’ reliance on communications bandwidth. If the robot can be treated as a trusted proxy for human scientists then we can send them to locations that would previously be considered unfeasible or wasteful.

The algorithms developed for space missions of course have applications here on Earth. Prospecting for resources is an important task whether it is in support of colonies on the Moon or in support of resource extraction for economic actors on Earth.

Humans Require Science Autonomy

The scientific method is a systematic process of generating quantifiable knowledge. Humans, who are riddled with cognitive biases, are a problematic component of this process. Confirmation bias, affirming the consequent, availability bias, and clustering illusions to name a few, interfere with human ability to conduct scientific inquiry.

The literature on the design of experiments, starting at least as early as Smith (1918), is an attempt build a mathematically rigorous approach to asking questions that ameliorates human biases. Science autonomy is another step along this path. An illuminating anecdote comes from Daston and Galison’s book Objectivity (Daston, 2007, pp.11-13) about Arthur Worthington’s research into the physics of how water droplets splatter when the hit a surface.

Worthington, through careful observation with his eye and a flashing light, documented how water droplets would split up upon hitting a hard surface. Circa 1897, he extensively documented symmetry and regularity in the patterns. The symmetry and beauty of his observations spoke to their necessary truth. What happened after he introduced photographic apparatus into his experiments warrants a protracted quotation from (Daston, 2007):

For years, Worthington had relied on the images left on his retina by the flash. Then, in spring 1894, he finally succeeded in stopping the droplet’s splash with a photograph. Symmetry shattered. Worthington said, “The first comment that any one would make is that the photographs, while they bear out the drawings in many
details, show greater irregularity than the drawings would have led one to expect.” But if the symmetrical drawings and the irregular shadow photographs clashed, one had to go. As Worthington told his London audience, brighter lights and faster plates offered “an objective view” of the splash, which he then had drawn and etched [...]. There was a shock in this new, imperfect nature, a sudden confrontation with the broken particularity of the phenomenon he had studied since 1875. Plunged into doubt, Worthington asked how it could have been that, for so many years, he had been depicting nothing but idealized mirages, however beautifully symmetrical.

Like Worthing we must ensure that observations are derived from meaningful, representative data. We must make sure that they are representing the true underlying statistics of the phenomenon we are concerned about and not imposing some preconceived notion of truth. If we cannot trust human perception to be unbiased, can we trust their decision making? Science autonomy can produce assistive tools for when our rationality fails us.

Science Autonomy is an Interesting Problem

Science autonomy is the mathematisation of the philosophy of science. Scientific inquiry presents an interesting set of behaviours that are designed to be rational and logical, but that need to be produced in a reasonable period of time given finite computing resources.

Science autonomy necessarily fuses reason and deduction with symbols grounded in the real world and in observable quantities. However, unlike other branches of robotics, it is strictly necessary that science autonomy return information that is human interpretable.

If we contrast science autonomy with other machine learning approaches, there is not much room to return, for example, a hyperparametric function approximation. In turn, this leads to interesting questions about what it means to understand the universe and what are the limits of human understanding. While this thesis does not address these questions, they do motivate the field in general.

1.2 Thesis Statement

The work presented in this thesis looks at designing algorithms for science autonomy, with particular consideration for operations in the field. They are all underpinned by the notion that by accounting for operational conditions the performance of science autonomy algorithms will be improved. We will consider this hypothesis in three settings: Opportunistic sampling of discrete objects (foraging), Opportunistic sampling in a field (prospecting), and Experiment design for hypothesis falsification (informative action planning).

We improve foraging algorithms by recognizing that agents do not have random access to all objects they would like to sample. We improve prospecting algorithms by recognizing that decision limits decided \textit{a priori} do not necessarily work for unknown environments. We used a decision making process that detects changes in distribution underlying observations with confidence, in order to make prospecting decisions.

We improve informative action planning by recognizing that scientists come to missions with pre-existing hypotheses, and we can use them to direct information gathering actions. Motivated
by the concept of falsification, we incorporate the hypotheses into the planning process.

The experiments in this thesis mirror settings from analog and planned science missions. We extend the state of the art in science autonomy algorithms by considering the missions constraints from field work.

1.2.1 Opportunistic Sampling of Discrete Objects (Foraging) - Chapter 3

Opportunistic sampling of discrete objects is perhaps best exemplified by robots like NOMAD and the OASIS/AEGIS system deployed on the Mars rovers. Robots identify something in the environment – a rock, a strong match with a template – and then decide to sample it. This is very much a Pavlovian, stimulus-response sort of behaviour, if an object of interest is identified, then sample it. The purpose of sampling is to determine some distribution underlying these classes of discrete objects, but what hypotheses are can be from collected observations are not considered.

![Image of a robot encountering discrete objects](image)

**Figure 1.2:** An example of a robot which encounters discrete objects in the environment. When the object, marked by the yellow circle, is far away it may not be identified by the perception system of the robot. As the robot gets closer to the object it can then be resolved by the perception system. Finally, as the robot passes the discrete object it no longer can affect the perception system. Images produced using the Resource Prospector simulator courtesy NASA Ames Intelligent Robotics Group.

The state of the art algorithms consider the value of these discrete objects to be binary, and they do not consider the result of sampling these objects. Additionally the state of the art algorithms in both science autonomy and statistical experiment design do not consider the distribution with which these sampling opportunities are encountered.

In this chapter we demonstrate that our algorithm, called Foraging, presents an improvement over the practice of always engaging with available samples, for large sampling costs. We also demonstrate that the Uniform sampling algorithm performs better than our proposed algorithm for some cost setting, but it also performs worse than both our algorithm and the algorithm which samples everything it encounters.

The proposed algorithm represents a compromise algorithm which has some of the improvement of the Uniform sampling strategy without losing the advantages of the strategy of always
sampling. We have identified different regimes of sampling and exploration costs which favour the different algorithms, knowledge which could be used to plan mission sampling strategies.

### 1.2.2 Opportunistic Sampling In a Scalar Field (Prospecting) - Chapter 4

Opportunistic sampling in a scalar field, or prospecting, is the navigation a space in order to determine locations considered valuable with respect to an objective function, illustrated in Figure 1.3. In this chapter we consider robots that deploy discrete actions in response to the readings from its proxy sensor. A Neutron Spectrometer, for example, is a proxy sensor for water deposits, and is slated to be part of the resource prospector mission (Andrews et al., 2014). The neutron spectrometer will report a reading at any point in the environment, and even subtle changes in the sensor position can change the readings of the sensor.

![Figure 1.3: This image illustrates the notion of prospecting. The transparent versions of the robot represent previous points in time. For every location the robot passes through the primary sensor gives a valid reading. The robot has to choose where to a sampling action in the field of observations.](image)

In practice, autonomous robots conduct prospecting by monitoring the readings from the prospecting sensor and if the reading crosses a threshold then deploy the sampling action. The threshold may be on the raw data being reported from the prospecting sensor or it may present in the form of a threshold in the surprise of a single reading, given a distribution. In either case, the response is to an immediate sensor reading, and not to the belief that a change in the underlying distribution has occurred.

Our approach is able to detect sub-threshold changes in sensor readings, and does not need to be tailored to every individual setting. We demonstrate on real-world data from the Mojave Volatiles Prospector (MVP) project that our algorithm is able to detect sub-threshold distribution
changes that are not detected using the threshold approach. The threshold used in our experiment was determined for this test site and the data were all collected in the same region. This illustrates the strength of our algorithm compared to the those used in practice.

The algorithm gives a measure of confidence in the belief that the underlying distribution has changed, which can be tailored to the risk posture of a mission. It could also let the autonomous scientist judge accordingly how risk adverse it would be, given mission resources. In principle the technique used in this chapter should translate to vector fields as well, but that application remains for future work.

1.2.3 Experiment Design for Hypothesis Falsification (Informative Experiment Planning) - Chapter 5

State of the art in informative path planning focusses on collecting data that can be used to estimate some unknown function, for example the spatial distribution of subsurface halite as a function of location in an environment. The paths are formulated to maximize information gained, for some measure of information gain. However, these approaches assume that the robots are engaged in exploration with no or one hypothesis being investigated as part of the mission.

In this chapter we examine the problem of designing experiments, and hence selecting samples points in 2D space, for a robot collecting samples which are designed to distinguish which of a set of hypotheses are most likely to be correct. The robot is considering multiple competing hypotheses about what it would observe at different locations in space, and it needs to determine sampling locations which have the greatest likelihood of distinguishing between hypotheses. This is an information gain problem, but instead of trying to maximize information gained in the distribution over the parameters of one hypothesis, we are attempting to maximize information gained in the belief over the hypotheses themselves.

Our approach is a sampling based approach that is agnostic of the form of the hypotheses under investigation. As with other sampling-based planning, the approach lends itself to designing experiments in 2d environments, but also in more abstract mathematical spaces.

We demonstrate that by being aware of the hypotheses under investigation we are able to determine paths that more readily identify which of a set of hypotheses best explain the data. We also demonstrated that when none of the hypotheses are “correct” (i.e. a probability of predicting the observations > 0.5) our proposed algorithm is better at predicting that none of the hypotheses should be considered.

1.3 Scope of Work

Science autonomy touches on many aspects of robotics and artificial intelligence. Consequently we feel it is important to clearly demarcate the scope of this work. The problem settings tackled in this thesis are narrowly focused on the author’s work at NASA Ames Intelligence Robotics Group, and hence

We do not consider perception as a problem in this thesis. This is, in itself, a wide field of work. We assume that some mechanism capable of identifying discrete objects in the world is available to our robot scientist. Template matching as been previously satisfactory for robot
scientists, and advances in machine learning for perception give us hope that even better systems may be available in the future. It would be beneficial, but sample-expensive, to use samples collected in a science mission to help train a perception system to distinguish meaningful features in sensor data.

We do not address efficient path planning. For the first two components of the work we assume that a trajectory has already been given to the robot and our algorithms are able to interrupt that trajectory. Opportunistic science algorithms which are path-agnostic may make less efficient decisions, but they can be more easily integrated into an existing robot. This situation aligns with anticipated operations in future NASA mission, where mission planners wish to determine long-range trajectories.

In the third component of this thesis we consider a greedy planner. Here our problem setting is with a landing vehicle that is capable of reignition and as such does not pass through every intermediate point in the map. While more efficient path planning would be beneficial, the main purpose of the chapter is to consider the difference between mutual-information sampling in the domain of the hypothesis and mutual information sampling in the belief space of a set of hypotheses. Additionally, as this algorithm also has applications to exploring domains where it is possible to have random access to elements of the domain of the hypotheses, path planning is not necessarily a meaningful concept.

We do not consider multiple communicating robot scientists, nor do we consider hypothesis generation, summarizing hypotheses and data for human consumption, or prioritizing data for downlink. We do highlight some approaches to these problems in chapter 2, but the topics themselves are outside of the scope of this thesis.

1.4 Summary

We view the scientific process as a collection of tools for designing experiments, conducting experiments, and using collected data to update or generate new hypotheses. We believe by accounting for operational conditions we can improve the performance of some of these tools. In this these we consider two reactive, opportunistic science algorithms, and a third, deliberative science algorithm which plans actions to determine which of a set of hypotheses are most likely to be correct.
Chapter 2

Related Work

Robots have been involved in collecting scientific information since at least the Voyager probes \cite{Krimigis1983}. Since then vehicles have developed increasing levels of autonomy, as exemplified by the robot Zoë from the Life in the Atacama Desert project \cite{Thompson2008}. In this chapter we consider robots that have been engaged in field work, specifically planetary science and underwater exploration.

We do wish to stress that while some of the approaches discussed in this chapter are considered more or less autonomous, this is not meant to be a value judgement of the work. Vehicles which ferry instruments in order to collect data, but make no decisions about which data to collect, can be vital components of an autonomous science mission.

First we briefly discuss robots which engage in relatively passive data collection. Next we review work relevant to this thesis, namely opportunistic sampling and informative action selection. Then we review hypothesis generation research, which motivates the falsification-based informative action selection developed in chapter 5.

2.1 Passive Sampling

The first robot scientists were simple probes that arrived at a location with scientific instruments, recorded data and reported those data to remote humans. They acted as ferries for scientific instruments, if any decisions were to be made, remote human scientists made them. The motivation for using robots was to eliminate the risks to human life that would be incurred by sending humans to explore, and the costs of bringing humans back.

Early examples of these robot scientists include NASA’s Voyager \cite{Krimigis1977} and Surveyor \cite{Ezell1988}, pp 325-331 probes, as well as the Venera \cite{Vakhnin1968} and Vega \cite{Sagdeev1986} programs. Even the modern Phoenix \cite{Arvidson2009} and Philae \cite{Hand2014} landers served the purpose of landing in one place and collecting what data it could there.

The Voyager spacecraft have collected data that would not be observable other than through their presence, e.g. \cite{Krimigis1983} or \cite{Krimigis1977}. The Vega landers, despite Vega 1 failing during landing and Vega 2 lasting only 56 minutes on the surface, returned information about the Venusian surface and atmosphere without any intelligence or autonomy...
(Bertaux et al. 1996; Surkov et al. 1986). Similarly the Viking landers revolutionized the understanding of the Martian surface and hydro-geological processes there (Raeburn, 1998).

These early space exploration missions do not necessarily fit the image one might have of a robot explorer. A vehicle that makes decisions about where it travels, where it collects samples, and what it does with those samples once collected seems more appropriate. However, when little is initially know, collecting any data is valuable. On an alien body one lander can reveal a wealth of information even with only a few images or spectra.

More recognizable as a robot, the Dante II robot (Bares and Wettergreen, 1999) demonstrated the operation of a robot collecting observations. Dante II is the poster child for robots solving the “dirty” and “dangerous” parts of the “dull, dirty, and dangerous” triad of robot tasks. Volcanologists have previously died collecting samples from Volcanoes. Dante II was able to operate with remote operators a safe distance from the actual data collection site. An early robot, it was not without faults and failures, but as a proof of concept it succeeded admirably.

Kunz et al. (2008) describes operations with two robots, Puma and Jaguar, which conducted undersea exploration. This work looked at operating under the ice sheet in the Arctic as an analogue for operations on Europa or Enceladus. There the vehicles collected conductivity, temperature, and depth readings while following a zig-zag pattern across the spatial extent of the area under the ice, as well as an oscillating depth profile while exploring. The autonomy of the system permitted the vehicle to follow the desired trajectory and collect the necessary data without human intervention.

Bingham et al. (2010) presents work on using robots to do deep water archaeology. This robot follows a lawnmower pattern and uses precise navigation information to co-register a suite of sensor readings to build an accurate map of underwater archaeological sites. Similarly, Furgale et al. (2010) presents an approach that follows transects specified by humans. It uses good motion estimates to register ground penetrating radar with the geometric model of the terrain. It also corrects for the topography of the terrain it is sensing, thus improving the sensor’s behaviour. While these robots do not make any decisions about their actions they nevertheless collects valuable and interesting scientific information.

The lawnmower, or boustrophedon, path (LaValle, 2006, §7.6, due to Acar and Choset), which these robots follow is a special case of a space filling curve. While there is a lot of effort in robotics in planning informative paths, space filling curves have the advantage of being to pass through arbitrary points in an n-dimensional volume, which means that uniform coverage can be guaranteed in navigable regions.

Spires and Goldsmith (1998) has conducted work into using space-filling curves for geographic exploration. They note some benefits of efficiency for multi-robot exploration. However one downside of typical space-filling curves which are not in the standard lawnmowing pattern, is that sometimes the robots make progress away from the desired goal. Depending on the probability of robot failures and the mission constraints on speed made good, this may not be an acceptable property. On the other hand, space-filling curves generalize to greater than two dimensions, making a planner more generalizable.

Uniform sampling is fundamental approach to scientific experiments. With sufficient sampling budget, sampling uniformly and frequently over the domain of interest is a standard way to collect data. However, in general spacing samples evenly along the length of a space-filling curve may not guarantee uniform sampling in the space the robot is navigating.
Following space-filling curves has its limitations. Choosing the resolution of the space-filling curve \textit{a priori} does not guarantee a good fit to the spatial resolution of the underlying phenomenon of interest. Space-filling curves may produce trajectories much longer than an informative path planner might. Informative path planners gain efficiency over the lawnmower pattern by adapting their sampling to minimize wasted resources.

Second, the space-filling curves generally assume free motion in a convex space. The pattern does not account for bottlenecks in terrain or non-uniform path costs. While these curves can be adapted to more complex terrain some manner of planner is still required, adding overhead to the robot.

Despite these shortcomings it is worth examining if space-filling curves can be used to satisfy mission requirements. In combination with opportunistic science a space filling curve can be more than adequate. Indeed, the lawnmower pattern is used to great effect in research such as \cite{jakuba2008} and \cite{yoerger2000}. It is still an active area of research, \cite{wilson2017}, have produced a new means for building a coverage for regions of interested defined by the contours of a Gaussian process regression model which the vehicle is learning.

This is a very small sampling of robotic platforms that have been used to collect scientific data. The important less to take here is that these vehicles have all provided useful data, and with a minimum of autonomy. Given the validation and verification overhead costs of flying autonomous systems, simple systems that follow space filling curves can be beneficial and inexpensive. When planning future missions it is important to consider the marginal costs and benefits of flying adaptive science robots.

While “simply” collecting data is the first instantiation of robot explorers it is by no means an outdated methodology. When one knows nothing about a new environment any data collected is extremely valuable, arriving in one location and collecting observations can be profoundly informative. However, if one has limited sampling resources, then one may need to consider more complex robots.

\section{Opportunistic Sampling}

A more complex autonomous scientist is one that can react opportunistically to the phenomena it observes. Plans generated before exploring a space necessarily cannot account for unexpected sampling opportunities. Because sampling opportunities encountered can be both unexpected and potentially scientifically valuable, a robot scientist must be able to choose to sample opportunistically.

In the previous section, it was assumed that sensors were always recording or effectively free to sample. Since there is no sampling decision to be made, opportunistic sampling is not required, as all observations will be recorded. However, if there are costs to collect samples or an instrument must be activated, then the robot needs to make a decision about how to spend what sampling resources it has. Because of this opportunistic sampling is a necessary component of a robot exploring an unknown environment, increasing the likelihood of making serendipitous discoveries.

Opportunistic sampling is decoupled from the trajectory planning process. It is a reactive
process that decides *in situ* whether or not to deploy sampling resources. Viewed through the window of design of experiment literature it has parallels with the Multi-Armed Bandit (Robbins, 1952) and Secretary Problem (Ferguson, 1989) literature. Multi-armed bandits are a formulation for addressing the exploration/exploitation problem. However, it makes the assumption that all of the actions are available at any given time, which is not necessarily the case when the actions are sampling objects distributed arbitrarily through the environment.

The secretary problem is a challenge to pick the best out of a sequence of actions, where the player can rank every action upon seeing it, but once it has passed up an action it cannot return to it. This formulation makes the assumption that the decision maker has perfect knowledge about the relative values of different sampling opportunities, which is not necessarily the case when exploring an environment with noisy observations.

### 2.2.1 Opportunistic Sampling of Discrete Objects (Foraging)

We first consider robots that opportunistically sample discrete objects. These are items that show up in the sensor data that can be segmented out from the surrounding region of the data stream, i.e. a rock in an image. Many of these approaches use template matching to identify targets, or variations on these themes such as SVMs or Bayes’ nets. The fundamental idea behind them is the same, however, is that an object is identified in the data stream, distinct from its surroundings, and then sampled.

The first robot to consider in opportunistic sampling is the NOMAD robot (Wagner et al., 2001). This robot explored the Canadian arctic looking for meteorites. While it was following its path it would search camera images for data that matched a visual template. If a template match was reached above a certain threshold then it was determined that something interesting was discovered, and NOMAD would collect a sample. However, the results of those samples are not used to determine the relevance of the templates used by the robot. NOMAD may be the first example of an autonomous robot scientist that makes decision in the field. Many of the robots we will discuss in this section use some form of template matching, the implications of which we will discuss below.

A slightly earlier paper, Gilmore et al. (2000) focusses more on identifying objects that warrant attention either from scientists or to be put into a downlink to human scientists. They use classifiers to determine what part of the sensor data represents an interesting object. The robot directs cameras for higher resolution images, but does not change the overall trajectory of the vehicle. The classifiers that are used are informed by mission-relevant constraints, but they don’t consider what happens if the classifiers are malformed. The system requires orbital imagery to be collected prior to mission deployment, which limits its ability to operate in totally alien worlds. While precursor data collection is happening on the Moon and Mars it might not always be feasible for first run missions or on underwater missions, or in lava tubes.

The Advanced Sciencecraft Experiment (ASE) software that runs on EO-1 identifies events of interest by using the multi-spectrum Hyperion instrument to identify targets of interest (Chien et al., 2003). This work is extended in (Chien et al., 2005), which presents software running on the EO-1 that is responsible for “science data analysis, mission planning, and run-time robust execution”.

Targets of interest are identified as one of: Thermal anomaly detection, cloud detection,
flood scenes, change detection, or other classifications based on on-board classifiers. Each of the classifiers have an associated score which is used to score potential targets. The classifiers and the relative scoring of outputs is pre-determined by scientists. This can be updated during the mission of the EO-1, but they are not modified by the results of observations by the ASE software.

Should identified targets be deemed interesting, and in alignment with scientist-specified mission objectives then the vehicle trajectory will be modified to collect further observations. The observation plans of the EO-1 are then modified to collect more observations of high-ranking targets and to remove planned observations of low-valued targets. Because the EO-1 is constantly orbiting the planet it can take advantage of multiple views of the same location to identify time-varying events in addition to any static features or characteristics that may be of interest to scientists.

The EO-1 employs a planner called CASPER which is capable of building and repairing plans (Chien et al., 2000). CASPER incorporates mission objectives and schedules actions in order satisfy mission objectives. CASPER performs plan repair in the case of conflicts between different objectives entered into the plan. CASPER is also used with the AEGIS software for the same purpose. Both ASE and AEGIS employ additional safety checks to ensure commanded actions are safe and viable. While safe-keeping autonomy is vital to autonomous operations, it appears that results of safe-keeping actions are not fed back into the science autonomy system. Employing the CASPER system with an opportunistic sampling algorithm is important, as robots can then negotiate the consequences of the selected sampling actions.

What the OASIS system is lacking is a trajectory planner. The CASPER system is discrete action scheduler, but since OASIS was developed in the context of the Mars missions, there was seldom need to develop long-range or over the horizon trajectories. There is nothing precluding the OASIS system from being integrated with a long-range planner, and in fact it would be quite a natural extension.

In the ASE software we see the framework of: Identify target, describe and rank targets, then collect follow up data with high-cost instrument. ASE can be viewed as a progenitor of the On-board Autonomous Science Investigation System (OASIS) and the Autonomous Exploration for Gathering Increase Science (AEGIS) system, which can be viewed as a particular instantiation of OASIS.

The OASIS/AEGIS system is currently in operation on Mars and is an excellent exemplar of an opportunistic science system. Successful fielded deployment of AEGIS is vitally important to recognize, as it underscores that autonomous systems can be trusted to make scientific decisions. Even though AEGIS was not given a longer leash until well into the life span of the MER rovers, it still represents a substantial investment and an impressive body of work.

OASIS, as described by Castano et al. (2007), uses three methods to identify science targets. The first is to match templates (as either weights on features or exemplar images) identified by scientists. The second is to use novelty detection in a feature space, which is done using three different modelling methods. The third is to use representative sampling, which ensures it sends examples of stimuli from all the clusters it has identified in scientist-defined feature space. The representative sampling is to ensure that rare objects (e.g. uncommon rock types) don’t get swamped out by an over abundance of any other types.

OASIS is a framework which has formed the basis of a number of different science autonomy
systems. For example, \cite{Gaines:2010} uses scientist-defined signatures (read: templates) to
determine what is of scientific value. Additionally, they use novelty detection to help prioritize
data for downlink to Earth. Their data prioritisation scheme comes from the OASIS system, it
combines novelty detection, signature analysis(templates), and extracting representative samples.
This does let the robot redirect itself to collect more science data. While it is an inherently remote
sensor, as it is a balloon collecting visual data, the framework for conducting exploration is still
valuable. OASIS does allow the robot to change the path and to override scheduled tasks, which
permits greater freedom for the vehicle than what was deployed on Mars in the form of AEGIS.

\cite{Estlin:2012} document AEGIS, which was uploaded to the Mars Exploration Rovers
(MER) in 2009. The purpose of AEGIS is to select potential science targets from sensor data that
may warrant follow-up activities. The objective of AEGIS is to use wide field-of-view (FOV)
sensors, like the navigation cameras on MER, to direct the use narrow FOV sensors, such as a
spectrometer or high magnification imager, and thereby collect information for scientific inquiry.

AEGIS executes at the end of a traverse and uses the navigation imagery to find sites of
potential interest. Should any be found the targets are imaged using the narrow FOV multi-spectral
panoramic imager onboard MER. Should no targets be identified the system takes no actions.
By pre-filtering the data in the navigation imagery and selecting relevant or interesting targets
AEGIS saves on bandwidth by only returning potentially interesting data to Earth. Additionally
not transmitting the original navigation imagery for scientists to identify science targets reduces
collection time by the time it would take to transmit the original imagery, have scientists identify
targets in the scene, determine best actions to take, and return commands to the rover in sync
with the next command cycle.

AEGIS processes imagery in the following pipeline: It identifies targets, extracts features
from the image, prioritizes the targets, determines where it needs to point the narrow FOV sen-
sors, the collects data with the narrow FOV sensors. AEGIS identifies targets by using an algo-

rithm called ROCKSTER which identifies rocks in a visible light intensity image. Once rocks are
identified they are described by the features of size, reflectance, shape, and location. Scientists
have pre-loaded templates that identify feature values of interest, given the local terrain. It re-
 mains unclear if the contextual information of “in the local terrain” is encoded in some way other
than the fact that the scientist have uploaded templates to be used in the current context. With
the rocks prioritized AEGIS identifies the location of the rock(s) of interest and begins collecting
information with the narrow FOV multi-spectral imager.

What we see in AEGIS is the general framework of:

1. Identify potential targets in primary sensor data - ROCKSTER turns intensity images into
   regions identified “rocks”.
2. Develop feature descriptions for the targets - AEGIS “rocks” described by shape, re-
   flectance, etc.
3. Rank targets by some objective function - “rocks” are scored by how well they match
   templates specified by scientists.
4. Observe targets with high-cost secondary instrument - Imagery collected by the MER
   multi-spectral imager.

This framework is inherited through OASIS from EO-1. The generality of this algorithm
means it is not strictly tied to visual data or to the multi-spectral imager, as noted in (Estlin et al., 2012). There are considerable computational constraints placed on AEGIS to fit inside the computing hardware of the MER rovers. Consequently it uses edge detection to find the rocks. That being said there has been continued work at JPL using the TextureCam (Thompson et al., 2012) algorithm, which employs a random forest classifier using low resources to effect a better classifier for a wider range of targets.

For scoring targets AEGIS considers two features extracted from the data, \( f_1, f_2 \), at a time. The target scoring algorithm, given by \( v = \alpha_1 f_1 + \beta \alpha_2 f_2 \), is defined by three variables, \( \alpha_1, \alpha_2, \beta \). \( \alpha_1 \) and \( \alpha_2 \) determine whether high or low feature values are preferred, and take values of either 1 or -1. \( \beta \) is in the range \([0, 1]\) and determines the relative weighting of the two features. There is the additional ability to exclude targets based on thresholded feature values, in order to remove potentially spurious targets.

While AEGIS’ scoring function is limited in how it can related targets of different features, i.e. it cannot favour middle values of features, only extremes, it should be taken not as a limitation of the AEGIS algorithm but of the computing environment on which it is deployed. What these objective functions do lack, however, is some grounding in the results of the follow-up sampling action. In defense of AEGIS, a robot equipped with a radiation tolerant computer is unlikely to be able to process sample data in a reasonable period of time. Nevertheless, the lack of closing the feedback loop from decision function to the results of the decision means there is a cap in the framework that should be filled with a machine learning approach.

AEGIS represents a substantial step forward in autonomous science because the system not only identifies and classifies sensor data, it makes decisions about what data to collect and has been deployed on another planetary body. The authors of (Estlin et al., 2012) identify several directions that they feel are valuable for improving AEGIS. Chief among them are planning with respect to resource allocation and planning for collecting observations that require direct contact and/or vehicle motion.

The ProViScout team led by Mark Woods developed an autonomous scientist in the papers (Shaw et al., 2007; Woods et al., 2008, 2009, 2011; Paar et al., 2013). Like the OASIS family of systems, it is a framework for autonomous science. ProViScout makes decisions about what targets to investigate and when it is safe, in the context of mission objectives, to deviate from the current plan of action. It is also capable of conducting plan repair to finish the mission, should activities selected opportunistically cause unforeseen delays in the plan.

Paar et al. (2013) reviews field experiments with the ProViScout and they combine human designed feature detection as well as anomaly detection. They address the problem of testing with limited sampling budgets, which they did in simulation, and the problem of field testing the science decision making algorithm (SARA), described in detail in (Woods et al., 2008, 2009). The behaviour of SARA in the field was tested with large sampling budget limitations, to evaluate the decision making process separate from cost constraints. In additions to any decisions made by the software, they also had standard science investigation procedures that were conducted at the end of any given transect.

The ProViScout system uses pre-defined science goals (Woods et al., 2011). It also relies on aerial data, which might not always be available in exploratory settings, i.e. lavatubes or underwater scenes. This system uses a contextual model, which is fairly unique in opportunistic sampling systems. Contrast this to AEGIS, which only engages in opportunistic science when
stationary. However, the contextual model used in (Paar et al., 2013; Woods et al., 2011) is a pre-specified set of parameters. The contextual model can be updated, but must be done by an outside agent (Woods et al., 2009).

Like OASIS/AEGIS, ProViScout also employs a classifier for science targets that scientists have pre-trained (Woods et al., 2011). The targets identified by SARA are assigned scores by scientists in order to direct the behaviour of the robot (Woods et al., 2009). The assumption underlying this decision is that the remote scientists know the relevance of the different classes of objects. Adapting the value of the different classes of objects in response to observations collected is something that must be done in the context of the hypotheses being investigated, but could be constructed easily in an information theoretic framework, something acknowledged by Woods et al. The plan repair component of ProViScout, analogous to OASIS’ CASPER, is documented in (Woods et al., 2008). The decision to engage in the opportunistic sampling action is given to the Timeline Validation Control and Repair (TVCR) planner (Shaw et al., 2007). The sampling opportunity is given its science value score as a priority, and then asks the planner to insert the activity into the plan. Once the plan has been made consistent with mission goals, the robot executes the new plan, which may contain the newly requested action. However, this decision doesn’t account for the likelihood of encountering any other such opportunities, only those which have already been scheduled.

Both the OASIS/AEGIS and ProViScout systems are very similar frameworks for conducting science autonomously. They both take a Pavlovian approach to making the sampling decision: If the object is in the scene, passes their respective prioritization schemes, then sample it. The questions that the systems do not ask is: what is the value of this sampling opportunity, relative to what is available in the environment? This particular question we address in chapter 3.

Using classifiers trained by scientists is an approach that is less likely to impart confirmation bias to the robot. By specifying what objects the scientists want to be identified and not how they want the algorithm to identify them they remove the influence of preconceived notions on different features. However, any trained classifier is limited by the expressiveness of its structure and the data it was trained on, so biases may still reside in the system.

Using a classifier for identifying interesting objects in the scene does let scientists determine which data are interesting, however, this excludes all stimuli that are not identified by the classifiers. This is an acceptable approach if the classifiers are reliable, but given that robots on planetary missions are exploring other planetary bodies, that assumption may not hold. It is valuable to be able to identify objects or clusters in sensor data which may relate to the hypotheses the scientists are investigating.

Wagstaff et al. (2013) presents a mechanism for identifying potentially interesting targets in situ and reporting them to scientists. Wagstaff et al. used an SVD based approach to identify stimuli that are novel, given previous experiences. Such an approach avoids the risk of bringing preconceived biases into object recognition and selection. More importantly DEMUD identifies the features that make the stimulus interesting. When something is considered uninteresting it is removed from the design matrix, and the process is repeated. In this way DEMUD builds a library of objects that reflect the statistics of the environment. The core of DEMUD is that it evaluates the singular values of the design matrix generated from all previously collected stimuli. DEMUD outperforms competing algorithms designed for novelty detection managing to find all
six of the candidate classes in the UCI glass data set (Frank and Asuncion, 2010).

DEMUD is obviously not the only algorithm that performs unsupervised learning, but it is important because it was designed to operate efficiently, making it amenable to execution on space rated hardware. DEMUD is important for learning things that are new or anomalous in the scene, but we also want to connect these features to a variable of interest, like the work in (Das et al., 2015) which is discussed in subsection 2.2.2. Combining these approaches would help improve the ability of robots to recognize and understand anomalous phenomena.

The approaches described above to opportunistic sampling of discrete objects share a common shortcoming. They make decisions to sample an object once it has been identified via some means of classification, and scored as passing a pre-determined threshold on value. The systems do not consider the relative value of sampling these classes of objects, nor do they consider the distribution of these objects about the environment, they simply respond with a sampling action once conditions have been met. We address these shortcoming in chapter 3 and show that an improvement in behaviour is possible.

2.2.2 Opportunistic Science in Fields - Prospecting

Distinct from opportunistic sampling of discrete objects is opportunistic sampling in fields, mathematical fields that map the robot’s location to a vector or scalar value. With discrete objects there are only certain locations where the robot can collect samples. When sampling in a field, any location the robot occupies is a viable sampling point, regardless of how valuable or interesting it is.

Robots operate in fields when their sensors continually give readings, like a Geiger counter might, and not when there is a discrete object unlike what may be identified by an image classifier. We again assume that the robot is following a predetermined path, and that it makes decisions in situ to engage in extra sampling activities.

Thompson et al. (2013) model time series data collected by a robot following a transect. The robot has a sensor that is collecting data at a fixed rate and the only control the robot has is to slow the vehicle to collect more observations. The robot also considers a limited sampling budget which can be deployed. The algorithm considers the degree to which the data has deviated from an assumption of stationarity. The data collected are fit using Gaussian process regression, with a non-stationary kernel. How far the data has deviated from being stationary is determined by how much one of the kernel parameters is increased during the data fitting process. The speed of the vehicle is controlled in proportion to that parameter.

Like the work discussed in (Thompson et al., 2013), Girdhar et al. (2012) modulates the speed of an underwater robot collecting data. The robot collects images with a camera that is taking pictures at a fixed rate, analyzes the scene, and if the scene is anomalous the vehicle slows down, collecting more images. Their algorithm builds a topic model using a Bag of Words of ORB features from the images, and uses that topic model to score how surprising the current scene is.

Girdhar et al. use the surprise of the scene, measured with respect to the topic model, to control the speed of the vehicle. Images are viewed as being distributions over the topics in the topic model, and surprise was measured by a symmetric KL divergence between the new image and the closest image previously observed. As the surprise increases, the speed of the vehicle was reduced in order to collect more data, and vice versa. With each image collected the topic
model was updated, so with more observations, the robot builds a better understanding of the world.

Girdhar continues development of the autonomous underwater explorer, in (Girdhar et al., 2013a) and (Girdhar and Dudek, 2016). In these two papers the topic modelling is not applied to the whole scene in the navigation cameras of the robot, but to sub-regions of the image. This way different parts of the image can be identified as novel and hence attract the attention of the robot. Instead of simply throttling the vehicle speed as in earlier work, the robot is now capable of following whatever novel stimuli it encounters.

It would seem beneficial to ground the topic models in the scenes to other phenomena of interest. Das et al. (2013) build feature descriptions of underwater environments and relates them to the density of life in water samples. In contrast, Girdhar et al., only use the topic models for detecting sensory novelty.

The work in (Das et al., 2013) was later extended in (Das et al., 2015), where they focused on improving repeated surveys through integration of previously collected data. The algorithm in (Das et al., 2015) is simultaneously attempting to learn the relationship between environmental features and the abundance of plankton discovered in water samples while maximizing the number of samples of plankton collected.

The submodular secretary problem is a variant on the secretary which permits collecting $k$ samples instead of just one (Bateni et al., 2010). It achieves this by dividing up the transect into $k$ segments, and executing a secretary problem algorithm in each one. They demonstrate a reduction in regret in this paper, with both pre-recorded data and from field operations. The basic secretary algorithm is to ignore the first $N/e$ sampling opportunities, and collect the first object with a value greater than or equal to the first $N/e$ objects. The secretary problem is very useful when the explorer has non-reusable resources, like the water samplers in (Das et al., 2015).

In (Das et al., 2015) they use the GP-UCB algorithm to score potential sampling opportunities. They apply the GP-UCB algorithm to score candidate sampling locations opportunistically, while they use the submodular secretary problem to determine when to collect samples. The variance driving sampling algorithm that they used as baseline algorithm had a higher regret in terms of quantity of plankton observed, it seems to have learned what appears to be a statistically indistinguishable model for predicting plankton abundance from environmental conditions.

Yoerger et al. (2007) uses the Automatic Benthic Explorer (ABE) to localize underwater hydrothermal plumes. They command the robot to follow a lawnmower pattern through the water, collecting measurements with sensors to identify chemicals ejected from hydrothermal vents. Once the vehicle has reached the end of the transect it plans new site visitations based on whether something meets a “revisitation” criterion or not. The scalar-valued revisitation value is not specified in this work, but it is a static threshold (Camilli et al., 2004). This approach does result in a lot of back tracking on the part of the robot, but can be acceptable when mission risk posture permits it. However, using a static threshold does not account for the noise in the sensors or variability in the underlying process. Further, if the quantity being examined is transient, waiting until later to sample it could mean missing collecting valuable observations.

Ferri et al. (2010) addresses the shortcomings of the previous work. Like (Yoerger et al., 2007) they complete a lawnmower pattern, identify chemical densities that relate to hydrothermal vents, then if the densities exceed a threshold they go back to those places and engage in searching activity to localise the hydrothermal vent. Ferri et al. (2010) has an adaptive threshold...
A mechanism based on how often it has crossed its original translating path, and how many spiral investigations it has conducted compared to the total recommended number of spirals. The suggested number of spirals is 5, as that worked out to be between 5 and 10% of the overall range of ABE at 50km. This approach is an improvement over (Yoerger et al., 2007), but it seems that (Ferri et al., 2010) could benefit further from techniques from secretary problem literature.

The approaches presented in (Girdhar et al., 2012) and (Thompson et al., 2013) are about continuous control of a vehicle in response to a change in the field values without and with budget constraints, respectively. Das et al. (2015), Yoerger et al. (2007), and Ferri et al. (2010) deploy discrete actions in response to a vector (Das et al., 2015) or scalar (Yoerger et al., 2007; Ferri et al., 2010) field. Das et al. (2015) relates the local observations to a secondary value through the GP-UCB algorithm in a way which is not done in (Girdhar et al., 2012) and that isn’t strictly necessary in (Thompson et al., 2013), (Yoerger et al., 2007), or (Ferri et al., 2010).

What is lacking from these algorithms is a notion of confidence in what they are observing. Girdhar et al. chases novelty, Thompson et al. does not consider the relative likelihood that the underlying distribution has changed from stationary to non-stationary, only that it has been fit by the model. The robot in (Yoerger et al., 2007) finishes the entire transect before identifying the opportunities for sampling, incurring expensive travel times, while potentially missing opportunities to sample time-varying phenomena. While an improvement, the strategy in (Ferri et al., 2010) operates on a static threshold based on instantaneous readings, which leaves the decision making process vulnerable to erroneous readings. Additionally, by using a threshold on readings it means that the vehicle can’t react to sub-threshold changes in observations. Using an approach that acted on belief that the underlying observations have changed could benefit these systems.

The approach in (Das et al., 2015) uses a secretary problem algorithm to collect water samples to simultaneously maximize the volume of plankton observed and learn the relationship between environmental characteristics and plankton abundance. Since their objective is to maximize the observation of plankton in the sample the use of the GP-UCB does encode a measure of uncertainty in the prediction, helping it to make safer observations, but it seems to inherently assume that the input observations, the environmental characteristics, are stable readings. Filtering those values, or building confidence that the underlying distribution has changed could be a beneficial filtering stage for the algorithm.

### 2.3 Informative Path Planning

We move on now to robots that actively plan their motions through the world instead of simply reacting to observations. The robots we review next are determining paths or trajectories that maximize the information gained about an environment. The primary focus of these algorithms is to build maps that relate the physical location of the robot to some measurable quantity.

Robots operating without infrastructure often need to conduct simultaneous localization and mapping (SLAM) in order to build accurate georegistered models of data. Stipulating that SLAM is an important aspect of robotics, and an active research topic, this thesis will not go into detailed analysis of SLAM, and interested readers are directed to (Aulinas et al., 2008). What is important to understand is that mapping enables robot scientists 1) to determine spatio-temporal relationships between observations, revealing patterns not otherwise visible, and 2) to determine
what regions of the world have not been explored so they may plan their future actions.

Frontier exploration (Yamauchi, 1997) depends heavily on the existence of maps, maps that record not only data where the observer has been, but that are capable of representing where the robot hasn’t explored. A robot engaged in frontier exploration drives towards regions of a map that it has not previously seen. The robot may prioritize or triage frontiers for exploration based on arbitrary rubrics – traverse cost, distance to a goal, map coverage/certainty, etc. – and then visit the frontiers to collect information (Tao et al., 2007).

Frontier exploration can be viewed as a special case of information gain planning. However, as the space being explored becomes increasingly abstract then identifying frontiers becomes increasingly difficult. Formal information gain planning superseding frontier exploration is noted in (Visser et al., 2007). Freda et al. (2009) uses information gain to map a configuration space. They use a frontier based exploration method to make sure the space is effectively explored. Wettergreen et al. (2014) seek out unexplained spectral signatures in satellite imagery, effectively exploring frontiers of the observed spectral space.

Frontier exploration is driven by an important point: One should sample where one is most uncertain about the state of the world. This is a concept which was outlined in Kristin Smith’s work (Smith, 1918), which pioneered the field of design of experiments, and it continues to remain relevant. In many ways frontier exploration is what all science exploration robots are doing. However, in stead of simply seeking frontiers in physical space, modern robot scientists are exploring frontiers in information.

(Thompson, 2008) and (Thompson et al., 2011) present an algorithm that plans paths through overhead multispectral imagery that collects a set of observations that maximize their informativeness measure. Thompson and Wettergreen measure the informativeness of a candidate site by the that would be information gained given all previous observations. The change in entropy is measured with respect to the covariance function of a Gaussian process regression over the satellite imagery. If the entropy function $H(X)$ is defined over a set of observations $X$, and the rover has to choose a subset of all possible of observations $Q$, and to collect actual observations $A$, then the rover in (Thompson, 2008) is trying to find the path that reaches its end goal location while minimizing the quantity $H(Q) - H(A)$. While data the rover actually encounters when it enters the location corresponding to $a \in A$ is not integrated into the future plans of the rover, as it continues its exploration it can adapt its path to acquire a representative set of data.

This work is expanded in (Wettergreen et al., 2014), where the robot seeks out unexplained spectral signatures in satellite imagery. The robot here uses a satellite map of spectral observations, it then moves to locations to collect surface observations in an attempt to determine what local mixture of materials explains the spectra observed from orbit. As the robot travels it builds up a database of spectral signatures which it then uses to try and “unmix” the satellite observations.

The pixels are unmixed by taking the database of spectral observations collected thus far, and identifying endmembers. Endmembers act, in effect, as a linear basis to represent the other as yet unsampled satellite pixels. Those with the highest residual error are considered difficult to explain given the current database of observations. The sites that are difficult to explain are the most interesting ones, and so after every ground observation the robot re-plans and identifies a new locations to visit and sample.

A work similar to (Wettergreen et al., 2014) is that of Girdhar et al. (2014). They take satellite
images, break each pixel down into a bag-of-words of feature descriptors, and learn a topic model for each of the pixels. Then the pixels are scored for topic perplexity, which is a measure of how well the individual descriptor of the pixel is explained by the topic model. The sites that represent the greatest perplexity are then visited and samples are collected.

The perplexity score of Girdhar and Dudek parallels the unmixing procedure of Wettergreen et al. The major difference between the two is that Wettergreen et al. attempts to plan a shortest path that maximizes the information gained, while Girdhar and Dudek let the robot follow an unconstrained path. Consequently their trajectories appear rather more idiosyncratic, but there is no reason their approach could not be used with a more conservative planner.

Going further from the work presented in (Wettergreen et al., 2014), Candela et al. (2017) changes the problem somewhat by explicitly accounting for a hypothesis about the geological units present in remote sensor data. The global map is divided up into regions, each of which are assumed to represent some geological unit. The robot then designs trajectories that will maximize the information gained in the probability of the geological unit assigned to a region, given the observations made within those regions. This work is important for carrying with it a hypothesis about the environment it is exploring, and planning in order to shore up support in that hypothesis. However, it does only consider one hypothesis (the region divisions) at a time.

Hollinger et al. (2013) plans informative paths using a Gaussian Process with a non-stationary kernel function. This approach permits modelling variable uncertainty in the object it is tracking, which is important for representing models that are not stationary. In a similar vein, (Hollinger and Sukhatme, 2014) uses a sampling based approach to determine trajectories for an unmanned water vehicle that maximizes information quality - a generic concept standing in for variance reduction or information gain.

Work by the Williams’ group at the University of Sydney, documented in (Bender et al., 2010), uses the results of in situ measurements to direct the actions of an AUV to adaptively map the spatial distribution of an underwater coral reef. Bender et al. (2010) represents the spatial distribution of the coral reef by using a Gaussian Process Classifier. The robot’s sensory data was classified into one of two classes (reef or sand), and sought out to investigate regions of the map that are ambiguous, most interesting locations being where \( p(\text{reef}) = p(\text{sand}) = 0.5 \). By investigating those sites they reduce the ambiguity in the map. They compare the behaviour of their algorithm to a robot following a lawnmower pattern, and find their proposed algorithm improves the quality of the map.

One downside of the approach detailed in (Bender et al., 2010) is that the Gaussian Process Classification does not admit entropy calculations without a computationally intensive Monte Carlo estimation. To this end it makes using approaches like mutual information or maximum entropy sampling challenging, especially with limited computing hardware. The reward function they do use, finding ambiguous points, when scaled up to arbitrary numbers of classes could be considered similar to the perplexity measure of interestingness used in (Girdhar et al., 2014).

The work by Charrow et al. (2015) also plans an information gathering path. The manage to extract significant speed up in planning time by virtue of using the Cauchy-Schwartz Quadratic Mutual Information metric (Principe, 2010) and by discretizing the world being explored into voxels. Through the use of these methods their informative path planner is found to be much faster than with the standard Shannon Information Gain, as used in most other works in this chapter, but still achieving the same objectives. Building upon Charrow et al., the work from
Tabib et al. (2016) hinges on the notion that given an object being observed the information gained from sensors observing the same object should be independent, and as such, additive. This means that the robot can efficiently plan for multiple passive sensors simultaneously. This is, to the best of the author’s knowledge, the first planning algorithm that address the multi-modal informative path planning in a principled way.

Also eschewing the more standard Shannon information gain, Miller et al. (2016) uses the expected value of the Fisher Information to determine points of interest. Like mutual information (Lindley, 1956), Fisher information is used as a score to select the most informative experiments. Their path planner produces smooth paths that maximize the number of high information value observations. Fisher information and the mutual information are intimately related through the curvature of mutual information, but they are not identical quantities. However, a rigorous comparison of the behaviour of robots maximizing mutual information and those maximizing Fisher information does not exist in the literature, to the best of the author’s knowledge. One would conjecture that they would produce similar behaviour, but without such evidence it is difficult to select one reward function over the other and this should be studied further.

Similarly, (Schwager et al., 2017) uses the gradient of mutual information as a reward function. However, this was done not because of a connection between mutual information and Fisher information. Schwager et al. (2017) they examine teams of exploring robots, and use the failures of team members as a way of encoding hazards into the map of interesting phenomena. Where robots fail, their sensors report no information, effectively zeroing out the gradient of information gain, causing other team members, which are following the gradient of information gain towards maxima, to avoid hazards encountered by less fortunate robots.

Das et al. (2013) takes an interesting approach to science navigation by considering a composition of functions. They learn one function which maps locations in space to characteristics the environment at those locations, and a second function which maps environmental characteristics to the quantity of interest, namely abundance of plankton in water samples. Like the work of Thompson et al., they use a Gaussian process to learn spatial model of environmental characteristics, but they also use a second Gaussian process regression that maps environmental features to plankton abundance.

The environmental characteristics are learned from features that describe data from cameras and other sensors, such as temperature and chemical density sensors. The robot collects water samples that which are then processed between surveys to train the second Gaussian process. After each sample is processed the robot plans new trajectories to locations that maximize the chance of discovering life. They use the unscented transform to propagate uncertainty in feature predictions through to uncertainty in life abundance predictions. This can then be used to estimate the distribution over the abundance of life over the entire map, which is then fed into a maximum information gain planner. While this particular implementation of their project does have the disadvantage of not being able to update the map of life abundance in situ, the approach could be used unmodified if on-board processing of water samples is possible.

The interesting part of this design is that, assuming the features are useful for predicting other scientific quantities, this approach could readily admit multi-modal sensing for scientific inquiry, simply by maintaining other Gaussian processes mapping the feature space to the observations of other instruments.

These algorithms were mainly focused on planning efficient trajectories that also increased
informativeness, by some measure of informativeness. Almost all of them rely on the information gain being submodular, meaning that greedy algorithms can approximate the best algorithms with some acceptable margin of error. The notable exception to challenge this approach is in \cite{Hollinger2014}. There are four candidate functions for information gain: Variance reduction, Shannon Mutual Information, Cauchy-Schwartz Quadratic Mutual Information, and Fisher Information. Shannon and Cauchy-Schwartz mutual information are intimately linked concepts, but it seems that Cauchy-Schwartz is faster to compute, and therefore favourable.

Variance reduction is an approach that has been used since at least the beginning of Design of Experiments literature \cite{Smith1918}, and is the concept underlying the class of Upper Confidence Bound algorithms \cite{Lai1985} used in the Multi-Armed Bandit literature. Further, variance has a reciprocal relationship with Fisher information, so reducing variance is equivalent to maximizing Fisher information. Connecting these concepts even further, Fisher information is proportional to the second derivative (or curvature) of the mutual information between two distributions \cite{Gouriou1995}. However, Fisher information does have a strong dependence on parameterization, which could make implementation of more general algorithms challenging.

Smith in his Ph.D. thesis \cite{Smith2007} had to consider navigating a grid world in order to deploy a Ultraviolet Fluorometer in order to detect colonization of materials by microbes. This work demonstrates that doing onboard data analysis can improve the decision making processes of robot explorers. The problem was solved as a POMDP, which let the robot reason about the effects of its actions on its state of knowledge while it was exploring. This approach is rare in that it reasons about the actual observations it may encounter while navigating, instead of just making sure it has observed a reasonable sampling of the surface. POMDPs are, unfortunately, computationally intensive, and do not scale well as the map size increases.

\cite{Choudhury2017} attempts to overcome the computational complexity involved in using a POMDP model by employing a reinforcement learning strategy. The objective is to produce a function which can predict the relative value of candidate sampling in order to pick the best one.

\cite{Choudhury2017} simulate different exploration tasks where a robot is tasked with maximizing the information gained about a hidden world map. Here the algorithm can use an oracle with access to true world knowledge to select actions during training. The current state of the robot’s world knowledge and the oracle’s decision is used to train a heuristic that approximates the optimal decisions without the overhead of the computation time.

There are limitations in that this algorithm is only as good as the data upon which it was trained. Nevertheless, a reinforcement learning strategy could be readily adapted to online performance, and it is a reasonable approach to approximating POMDP based solutions.

\cite{Arora2018} details an algorithm for a robot that is seeking to locate a subsurface distribution of water. The robot here has to plan to not only trajectories that inform the distribution of subsurface water, but it also has to determine which of a set of instruments it will employ to collect observations. These instruments have different costs, and the observations may change the trajectory that may be the most informative. Their planning problem is complicated by not obeying sublinearity. They use approximate planning measures to deal with the complexity of the planning problem, and the use of an on-board robot-learned hypothesis about the relationship between the navigation sensors and the instruments used to determine the abundance of
subsurface water.

The work of Choudhury et al. (2017) and Arora et al. (2018) both attempt to mitigate computational complexity. The first through approximating the reward function, and the second through approximating the exact solution. This can be computationally crippling, but it is possible to mitigate it through approximate planning. This represents the next logical step in the state of the art, simultaneously exploring and learning hypotheses. Followed by simultaneous exploration and hypothesis generation.

What is common among all the information gathering algorithms presented above is that they are attempting to plan trajectories are in some measure the most informative about the environment they are exploring. With the exceptions of Candela et al. (2017) and Das et al. (2013), these algorithms do not consider hypotheses about the phenomena they are exploring, only efficient ways to explore the environment. Even (Candela et al., 2017) and (Das et al., 2013) consider one hypothesis at a time. In the next section we discuss different approaches to generating hypotheses, an important component of the scientific process, but if we are to combine informative path planning and hypothesis generation we will need a class of planners which are aware of and capable of reasoning about multiple hypotheses.

2.4 Hypothesis Generation

A great deal of excellent work in autonomous science research goes into ensuring that useful information is collected in a principled way, as per the work reviewed above, what is often lacking is the question of what is done with the data after the collection. In flight missions, scientists would use these data to generate hypotheses.

Hypotheses are an important part of conducting science, but it is not an active part of autonomous field science. While this thesis does not investigate the automatic generation of hypotheses, this is an interesting area of research that is relevant to the work conducted in chapter 5. Further, since chapter 5 is about designing experiments to test hypotheses, it is worthwhile being aware of mechanisms for generating hypotheses.

Levin introduced the notion of searching for Turing machines, which are functions that map input bit strings to output bit strings (Levin, 1973, 1984). Given a fixed language for describing Turing machines and a problem to solved the simplest machine could be discovered by first searching through all the shorter machines. Each one of the candidate Turing machines can be viewed as hypotheses that need to be tested to see if the satisfy the relationship under investigation.

The search space can be quite large, but the algorithm that successfully completes the task will eventually be found. Levin’s universal search procedure only considers functions that map from exact inputs to exact outputs, however, one could simply place tolerable level of error on the output and stop searching once that tolerance has been met.

Langley et al. (1987) presented a series of algorithms which discover empirical laws relating different datasets, as well as semantic rules for relating data that has been collected by some other process. Their work covers four algorithms, BACON, GLAUBER, STAHL, and DALTON. These algorithms deal with using data for finding quantitative laws, qualitative laws, inferring components of substances, and formulating structural models. In many respects algorithms
presented in that book are all types of search through a hypothesis space, and is an important precursor work to science autonomy algorithms presented below.

Levin’s search was further developed by Schmidhuber in [Schmidhuber, 1995] and [Wierring and Schmidhuber, 1996] for (single layer) neural networks and POMDPs, respectively. Again, the process is to start with simple attempts at solutions to the problem and progressively make them more complex until a satisfactory solution is found. Here the algorithm depends on a vocabulary of symbols that can be used to produce the hypothesized solutions. Each of the solutions can then be tested for fitness – how accurately the solve the problem – and then either discarded or retained accordingly.

The robots Adam and Eve were developed at Aberystwyth University to automatically conduct laboratory experiments [King et al., 2004, 2009a; Qi et al., 2010]. These robots automate much of the tedious work in laboratories, and help reduce variability in production. Adam generated functional genomics hypotheses about yeast, and tested the hypotheses in a laboratory [King et al., 2009b]. Adam conducted this work by exhaustively testing different hypotheses it generated about the yeast.

Eve, on the other hand, methodically screens candidate drugs, but is able to stop searching the space of hypothesized drugs in order to generate quantitative structure-activity relationship (QSAR) models [Qi et al., 2010]. QSAR modelling is a mechanism for determining the predictive power of the drug components to the potency of the drug. With a completed QSAR the robot can then predict drugs with similar structures which may be effective. This permits the robot to design new hypotheses about drugs which may be effective in treating the specified disease. This gives the robot the freedom to follow up on promising drugs instead of having to wait to finish testing all possible drugs in the space of drugs it is testing [Sparkes et al., 2010].

Schmidt and Lipson (2009) developed an algorithm for deriving natural laws based around symbolic regression (e.g. [Cramer, 1985]) which has since been developed into the product Eurequa. This approach relies on being able to represent equations and relationships as parse trees in a grammar over possible equations. They then apply a genetic algorithm to search the space of parse trees in order to determine a hypothesis which best fits the data that has been collected.

Because this approach is not searching progressively through the space of possible hypotheses in the same way that a Levin search does there is a need to control for over fitting. The approach used by Eureqa is to score each hypothesis based on its fitness to the data set under inquiry combined with a penalty term for the computational complexity of that hypothesis. In [Ly and Lipson, 2012] they use the Akaike information criterion (AIC) which is defined in Equation 2.1.

$$\text{AIC}(h, D) = 2|h| - 2 \ln(\mathbb{P}(D|h))$$ (2.1)

Where $|h|$ is the computational complexity of the hypothesis being scored and $\ln(\mathbb{P}(D|h))$ is the log likelihood of the data, $D$, given the hypothesis, $h$. Smaller values of AIC are preferred. The search algorithm that describes this system could modelled as in [Algorithm 2.1] where the generate_hypotheses method is the symbolic regression algorithm described in [Ly and Lipson, 2012].

www.nutonian.com
Algorithm 2.1 A skeleton of the hypothesis search algorithm used in Eurequa. hypothesis generation is their symbolic regression function, and the score function is the Akaike information criterion.

```plaintext
function SEARCH_HYPOTHESES(grammar,D)
    H ← ∅
    scores ← ∅
    while not terminated do
        H′ ← generate_hypotheses(H, scores, grammar)
        scores ← score(h, D) ∀ h ∈ H′
        H ← H′
    end while
    return argmax score(h, D)  
end function
```

The system that supports Eureqa assumes that the data to be fit has already been collected. It then uses the genetic algorithm to search through the space of possible hypotheses which are expressed in a grammar given by a vocabulary and set of operations specified by the user. This is advantageous as the system has managed to re-discover physical laws from data that are in a human-interpretable format, as opposed to, e.g. a neural network.

Genetic algorithms have the advantage of biasing the search space towards members of the space that have already been successful, and hence prune off a number of hypotheses that are unlikely to be productive. However, there is always the risk that the “best” hypothesis will never be evaluated, although this risk reduces the more generations are bred in the learning process. A similar approach to searching the hypothesis space is embodied in the Automatic Statistician.

This is a work that has been developed in three papers, (Grosse et al., 2012, Duvenaud et al., 2013, Lloyd et al., 2014). The program is designed to study a dataset and develop descriptions of the data in human-interpretable format. With the automatic statistician the hypotheses are the symbolic representations of the kernel functions, and it attempts to find one that best explains the data.

The algorithm works by generating covariance functions for Gaussian process regression and classification models and then translating those into natural language statements about the data. The method for generating the kernel functions is to start with a simple function and progressively make it more complex, as in Levin search, by iteratively applying the rules of the grammar to generate new and more complex functions.

The automatic statistician assumes that all data are available initially and exhaustively searches through the space of possible hypotheses to find the best one. They score their hypotheses using the Bayesian Information Criterion (BIC) (Schwarz et al., 1978), given in Equation 2.2, which is very similar to the AIC used in Eureqa, however it more sharply penalizes the complexity of the model. Where \( h \) is the hypothesis, \( D \) the dataset, \(|h|\) is the complexity of the hypothesis, and \(|D|\) the number of points in the dataset.

\[
BIC(h, D) = |h| \ln(|D|) - 2 \ast \ln(\mathbb{P}(D|h))
\]  

(2.2)

https://www.automaticstatistician.com

28
Like with Eureqa there isn’t an explicit stopping criteria, although the authors of the automatic statistician do put a limit on how deep in the tree of possible kernel functions it is permitted to search. With arbitrary complexity a model can be made to fit data arbitrarily well – overfitting – so there needs to be a way to trade off the fitness of the hypothesis with the complexity of the hypothesis. Where the Eureqa uses the AIC, the Automatic Statistician uses the BIC. We sketch the automatic statistician’s algorithm in [Algorithm 2.2].

**Algorithm 2.2** A skeleton of the hypothesis search algorithm used in the automatic statistician. The generate_hypothesis function is a Levin search through the space of kernel functions, and the score function is the Bayesian Information Criterion.

```plaintext
function SEARCH_HYPOTHESES(grammar,vocabulary,verbs,max depth,D)
    H ← ∅
    while depth not reached do
        H ← grammar-expand( H, D, vocabulary, verbs, depth, )
    end while
    return argmax h∈H score(h, D)
end function
```

Where Eureqa uses a guided search, the Automatic Statistician uses a depth-bound Levin search through a grammar of kernel functions, but ultimately they both attempt to find a hypothesis in the space of hypotheses which best fits a dataset, according to a fitness function that trades off performance with algorithmic complexity. This work seeks to address the fact that neither the automatic statistician nor Eureqa have a mechanism for recognizing that a new hypothesis is required, they simply search until they have found the best one. Nor do they determine where to collect samples to further improve their understanding of the hypotheses under consideration.

In the closing remarks of [Schmidhuber, 1995] the authors make the observation that true generalization is impossible. That any one model fit to data only really speaks about the data collected and can’t necessarily be trusted to predict on data that has not been previously observed. In many respects this reflects Hume’s problem of induction [Vickers, 2016] - any hypothesis cannot be truly trusted to predict outputs from inputs it has not been developed from. The logical problem of induction is what drives Popper’s falsification criterion and what also drives scientists to collect more data in order to test out their hypotheses, and when they find them wanting, generate new ones.

An ad hoc hypothesis is a modification to a favourite hypothesis to save it from being falsified in the face of new data. While the phrase “ad hoc hypotheses” is often considered a disparaging remark, it is important to remember that all hypotheses are ad hoc responses to new data that could not previously be explained, what separates “good” hypotheses from “bad” hypotheses is that the modifications do not reduce the predictive ability of the hypotheses and that those hypotheses are not later themselves falsified.

It is the need to continually falsify hypotheses that drives the notion of active learning. Agents need to collect more data in order to determine which of a set of possible hypotheses is the “best” hypothesis. Here best will be taken to mean maximizing or minimizing some objective function over the hypotheses and the data to be explained by them. Unless a very comprehensive set of
hypotheses are provided from the outset, an agent may well have to generate new hypotheses to explain the data they collect in pursuit of their scheme of falsification.

What all these approaches have in common is that they are using a grammar to generate a (possibly infinite) space of potential hypotheses, and they operate on a fixed dataset. They search the hypothesis space, either in a breadth first search, like (Levin [1973]) and (Schmidhuber [1995]), using a genetic algorithm to perform the search like the Eureqa project, or exhaustively with an imposed limit like the automatic statistician. These hypothesis generation algorithms do not consider, once a set of credible hypotheses have been identified, how to expand the dataset in order to determine which of those hypotheses are most credible. This problem is addressed by chapter 5.

2.5 Summary

In opportunistic sampling of discrete objects we have seen that the state of the art algorithms ignore the relative availability of different classes of objects. While they do evaluate taking opportunistic sampling actions with respect to the effect on the overall mission, they do not consider the likelihood of encountering more valuable sampling opportunities. We modify that short coming in chapter 3.

With prospecting algorithms we see that they don’t consider with what confidence a triggering event has been observed. We address that shortcoming in chapter 4 in order to demonstrate an improved ability to deploy secondary sampling actions.

Finally, when it comes to planning for information gathering we see two things. The first is that the vast majority of informative planning algorithms consider at most one hypothesis when planning paths and sampling actions for autonomous robots. The second thing we see is that algorithms which generate hypotheses only consider fixed datasets at the moment of generation.

What is not answered by these bodies of work is how to plan actions that are informative, in order to determine the most accurate hypothesis. In chapter 5 we develop an algorithm that plans informative actions in order to determine which of a set of hypotheses most accurately describes collected data. This kind of planning will be a fundamental component of robots which are simultaneously attempting to generate and choose between multiple competing hypotheses.
Chapter 3

Opportunistic Sampling of Discrete Objects (Foraging)

While exploring unknown environments robots will necessarily encounter unanticipated phenomena. To handle this situation robots require some mechanism that lets them react opportunistically to those phenomena. In this chapter we consider robots that are encountering discrete objects and that are interested in learning the underlying distributions of observations from sampling these discrete objects.

The object classes and the underlying distributions could represent many different things. For example, it could be types of drugs and expected outcomes, or, continuing a medical theme, it could be patients representing clusters of symptoms and recommended treatment. It could be classifiers acting on a data stream, and the underlying distribution could be its accuracy, as verified by some oracle.

To keep the project grounded in planetary exploration, we consider different classes of geologic materials, in particular, rocks, as the discrete objects, and the underlying distribution would be over the colonization by microbes of that class of rocks. We assume that the robot can identify the different objects with some inexpensive proxy sensor, but it can only collect observations informing microbial colonization by using a more expensive sensor. The overall goal of the robot is to estimate the distributions underlying the different classes of objects with minimal error.

We assume that the robot has no global information. That is, the robot does not know how many objects it will encounter, nor what their classes will be. We assume that the robot is following some pre-determined trajectory, such as a lawn mower pattern as in Figure 3.1.

The classes of objects that the robot encounters are drawn independently from a distribution representing the prevalence of different classes of objects in that environment. In each encounter with an object the robot can either choose to sample or continue exploring along the trajectory. Thus, the problem can be thought of as a stream of sensing opportunities, providing varying reward, and each requiring a decision to either sample or move on.

The state of the art algorithms in robotics literature for opportunistic sampling represent a very Pavlovian approach to opportunistic sampling. They generally either match a template or they respond to the novelty of the class of object. Seeking novelty is beneficial from both an information theoretic and a logistical approach. If class of object is rare, either it will provide greater information gain, because the explorer is unlikely to have collected many samples from
Figure 3.1: A cartoon of a path explored by a rover. The images represent different classes of desert pavements that may encountered by a rover as it follows a pre-determined path.
it, or logistically because there may not be many remaining opportunities to sample that class of object, so passing up the object doesn’t make sense.

A template matching approach, however, is more problematic. When we say template matching, we should note that we include any pre-trained classifiers in that phrase. On the one hand classification schemes are an efficient way for scientists to communicate their preferences for sampling to robots. On the other hand, if these templates are not properly vetted then template matching is a means by which confirmation bias can be encoded into the robot. That, in turn, can reduce the quality of the science conducted.

Both novelty seeking and template matching ignore the results of their sampling actions. That is to say, how much information was gained by the most recent sample of an object of a given class. There are standard tools in the design of experiments literature which do address this shortcoming. However, those algorithms, in turn, make assumptions which we will discuss in section 3.1 that do not hold in field operations.

Another consideration is that the number of samples that a planetary robot might carry. The Phoenix lander carried four copies of the single-use Wet Chemistry Laboratory (WCL) instruments Hecht et al. (2009). Zoë, in the second Life in the Atacama Desert project had 20 sample receptacles it could use to collect soil samples Paulsen et al. (2013). The upcoming Resource Prospector mission will likely be only collecting on the order of tens of samples with its drill Andrews et al. (2014). In planetary exploration missions sampling resources can be very limited, and as such one wants to be effective with small total sampling budgets.

In this chapter we introduce an algorithm to address the shortcomings of the novelty-seeking and template matching algorithms. The method we propose makes two major modifications to the state of the art. First, the objects are valued by the expected information gained by sampling the object. Second, we compare the anticipated information gain from the current object to the expected information gain for the next encountered object. With these improvements our algorithm is able to make the decision to either sample the object that is immediately available to the robot, or to continue along the trajectory in the hopes of finding a more informative opportunity for sampling.

The proposed algorithm draws on techniques from optimal foraging theory and sequential experiment selection. Its use is motivated by observations of human and animal behavior, exemplified by geologists making decisions about investigating local phenomena without prior access to detailed maps, who are able to effectively choose between sampling materials in front of them or moving on to potentially more profitable sampling locations. These decisions may not be globally optimal, but they offer robots a mechanism for making foraging decisions (either to engage with the environment or to continue exploring) in a computationally tractable way.

The algorithm is then extended to address another common situation in scientific exploration, namely environmental variability. As the robot traverses the environment, it may easily pass between regions where environmental conditions differ, which affects the distribution underlying the classes of objects. In the Atacama desert 100% of photosynthesis-promoting translucent rocks are colonized by microbes in semi-arid regions, but less than 50% of such rocks in arid regions, and less than 1% in the hyper-arid core Warren-Rhodes et al. 2007, 2006. If one can detect changes in the observed quantity, then one might infer changes in unobserved quantities in the environment, and trigger other investigative actions. Reacting to such changes is also relevant for information gathering purposes, so sampling decisions will not be based on historically ob-
served but now inaccurate class information. The proposed extension incorporates an additional statistical test to detect a change in class distribution to notify operators, separate data segments, and reset the observation history that might otherwise misinform upcoming sampling decisions.

To demonstrate the value of the algorithm we conduct a number of simulation experiments. First we give the robot a problem with three classes of objects to investigate. Next we explore the effect of the arrival distribution on the behaviour of the foraging algorithm relative to baseline algorithms, and the effect of the underlying distribution on the algorithms.

In all experiments we vary the exploration and sampling costs for the robot and non-uniform arrival probabilities. In the first experiment we consider a uniform arrival distribution with a small number of class of objects. In experiments 2, 3, and 4 we consider two arrival distributions which are heavily skewed in favour of one class of object over the others. We also vary the underlying distribution of the most common algorithm (experiment 3) in order to determine the effect of the algorithms’ prior belief in the likelihood of colonization on performance.

In experiment 5 we test the change-detection extension to the algorithm, by changing the distribution underlying the classes of objects. Here we select one sampling and exploration cost, and make them small relative to the overall sampling budget, in order to see how detecting changes in the underlying distribution can be used to improve estimates of class distribution parameters.

The remainder of this chapter begins with a brief survey of the relevant literature. Next, a detailed comparison of the proposed foraging algorithm and one based upon existing principles from the design of experiments literature. Finally, discussion of experimental results from a simulated exploration scenario indicates that under limitations on sample collection and overall mission time, the foraging algorithm presents a statistically significant improvement for a realistic range of sampling costs.

3.1 Prior Work

Automating experiment design and selection is not without precedent. Kristine Smith started the field of optimal experiment design in 1918 (Smith, 1918). Recently robots have been employed to conduct scientific inquiry autonomously (Wagner et al., 2001; Castano et al., 2007; King et al., 2004). Current robot scientists’ reliance on global information makes operating in truly unknown environments challenging. Additionally, previous approaches in sequential decision making from statistics do not necessarily reflect the settings that autonomous robots encounter in the real world. The particular approaches in question are formulations of sequential experiment selection problems, which are the Secretary Problem and the Multi-armed Bandit (MAB).

3.1.1 The Secretary Problem

The secretary problem asks a decision maker to select the best candidate from sequentially presented candidates where it is not possible to return to rejected candidates. In the original setting, there is only one position for the candidate to fill, (Ferguson, 1989) and the optimal strategy is to reject the first $\frac{N}{e}$ candidates and then accept the first candidate who is ranked better than any of the previously seen candidates. Further, the decision maker was able to objectively score the
candidates without cost. In our setting, we do not know the value until sampling, and sampling a class incurs a sampling cost.

There have been many variations on this problem including selecting multiple candidates (Vanderbei, 1980), or when the total number of candidates is random (Presman and Sonin, 1972), for more variations (Ferguson, 1989) is an excellent source. The Odds algorithm has been designed as an optimal solution to the secretary problem, but it requires knowledge of how many opportunities there are to collect samples (Bruss et al., 2000). More recently, the submodular secretary problem has been developed to handle selection of \( k \) candidates, as opposed to just one (Bateni et al., 2010). It simply divides the transect into \( k \) contiguous regions and runs the standard secretary algorithm on each segment.

What distinguishes the secretary problem from the particular science autonomy problem we propose is that we do not know the value of a candidate, or class, when we encounter it, it must be sampled to learn its true value. Additionally repeatedly sampling the same class decreases the value to the decision maker, whereas in the secretary problem the encountered value of the object is the value of the object. Since most of the secretary problems rely on previous observations of candidate values, our expected decrease in candidate value would not be compatible.

### 3.1.2 Multi-armed bandits

Sequential experiment selection, a type of active learning, is addressed in the multi-armed bandit (MAB) literature. This was introduced by Robbins (1952) as a means of sequentially selecting which experiments to conduct with a limited budget. In Robbins’ work, selecting experiments is modeled on determining the payouts of one-armed bandit machines – each machine representing a different experiment. The player has a fixed sampling budget and has to sequentially choose which machine to play, trading off exploiting expected rewards from well-studied arms against exploring different arms, learning more accurately the payouts of those arms.

Lai and Robbins (1985) use a value function in which uncertainty in arm rewards makes an arm more interesting. Recently decision rules like Thompson sampling (Thompson, 1933) and Bayesian Optimal Control (Ortega and Braun, 2010) have gained popularity. Other techniques addressing the exploration/exploitation problem use uncertainty as a reward metric (Burnetas and Katehakis, 1997; Auer, 2003; Balcan et al., 2006). In our setting, because the agent only needs to learn the distribution and not use it for anything, uncertainty is the only necessary reward.

Balcan et al. (2006) presents a method for learning classifiers by requesting samples from the input space with the greatest classification error. Classification error and uncertainty in function value are fungible quantities in this case. An analogy can be drawn between the classifiers used in (Balcan et al., 2006) and the bandit arms used by Auer and Ortner (2010).

Several factors distinguish the MAB setting from the problem explored in this chapter. In MAB, the agent has access to any arm (analogous to a class in our setting) it chooses at any given time. The agent in our setting does not get to choose which of the classes it can investigate. Any previously seen classes are no longer available, and new classes arrive per a random model. Additionally, the standard MAB setting does not have switching costs, although there are some formulations which do include such costs (Jun, 2004). In our setting, there is a cost incurred with every choice to continue exploring.
3.1.3 Optimal Foraging

Foraging is the problem encountered by animals seeking to maximize energy intake when operating in unknown environments. The central question of the problem is whether it is more valuable to continue extracting resources from the current location than it is to seek out resources in new locations. Charnov (1976) introduced a technique for dealing with “patchy” environments, in which there are distinct regions that contain different classes of resources. The forager can extract value from these patches, with diminishing returns (modeling resources consumed), or choose to continue to wander randomly in the hopes of encountering more valuable locations.

The optimal time to leave a patch, according to Charnov’s Marginal Value Theorem, is when the expected return from continuing to sample a patch is less than the expected return from searching the environment. In this formulation, the expected return from both the current patch and the environment are offset by the cost of extracting resources in this patch and the energy spent seeking a new patch.

Pirolli and Card (1999) studied researchers attempting to acquire information. They modeled the rate of information gain and had their agent decide to leave a patch when the rate of information gain was lower than that of the environment. What differentiates their setting from ours is that their decision maker can choose which patch to sample, yet our exploring agent cannot.

Kolling et al. (2012) studied humans engaged in a gambling task in which players have to consider the option they have before them and the opportunities the environment provides. Subjects were repeatedly presented with a choice of playing a gambling game or being randomly presented with a different game. Each game was a Bernoulli trial with some unknown probability of success. Kolling et al. identify possible neural substrates for foraging decisions in humans. The behaviour was near optimal, with some skewing of probabilities near 0 or 1.

3.1.4 Opportunistic Science

In chapter 2, we discussed the NOMAD (Wagner et al., 2001), OASIS/AEGIS (Castano et al., 2007; Estlin et al., 2012), and ProViScout (Paar et al., 2013) systems for deploying samples opportunistically. These algorithms use classifiers to identify objects in the scene, which are then prioritized via value functions specified by remote scientists. Any identified objects which meet a criterion for being sampled are then passed on to their scheduler to be integrated into the mission plan. We plan to show in this chapter that by recognizing the availability of the different classes of objects under investigation in the environment, that we can improve the estimate of the underlying distributions.

Das et al. (2015) present a method for deploying samples as robots explore along a transect, as does Girdhar et al. (2013b) and Thompson et al. (2013). However, the problem settings in these papers have robots operating in either scalar or vector fields, as opposed to sampling discrete objects. We will discuss these works in more detail in chapter 4.

We do learn from other autonomous science work that information gain, either in terms of Shannon (Thompson, 2008), Cauchy-Schwartz Quadratic (Tabib et al., 2016), or Fisher (Miller et al., 2016), is a useful quantity to evaluate the relative worth of different sampling actions. In this chapter we used the Shannon definition of information gain, however, in future work the
authors would recommend using the Cauchy-Schwartz Quadratic mutual information, if for no other reason than the decrease computation time noted by Charrow et al. (2015).

Sun et al. (2011) proposed a method for maximizing the expected information gain of a sequence of actions in a Markovian world. This work can be viewed as a more general formulation of the solution presented in this work. It forecasts out the actions of the robot to either a fixed or infinite horizon possible sampling opportunities. Our work can be viewed as employing the model developed by Sun et al. (2011) with a single time-step horizon, something more amenable to operation in restricted computing environments.

Previous work by the author employing optimal foraging techniques for science autonomy has considered robots with sampling budgets limited by a number of containers and assumed knowledge of the number of sampling opportunities that would occur Furlong and Wettergreen (2014b,a). While the limited sampling budget is realistic, foreknowledge of the transect is not necessarily so. This chapter improves upon the prior work by using productivity, the ratio of information gained to resources expended, to reason about sampling choices and gives a constraint of time instead of an unknowable number of sampling opportunities. In those works we found that algorithms which simply engage with every available opportunity never outperform uniform and foraging sampling algorithms, for that reason we do not consider them in this chapter.

3.2 Method

We consider a scenario where a rover is following a path set for it by remote scientists. The robot has a budget of 100 units of time which it can expend while following the trajectory. While following this path the rover will repeatedly encounter objects that belong to one of K possible classes. Initially the robot does not know how many different types of materials it may encounter.

At every encounter the robot has a choice of sampling that object, represented by taking action \( x \in X \) and making an observation \( Z \), or continuing along the path in the hopes of finding a more interesting sampling opportunity. The role of the agent is to determine \( P(Z|x) = \theta_x (1 - \theta_x)^{1-z} \forall x \in X \).

The experimental setup is a variation on Charnov’s patchy foraging (see Section 3.1.3). In this case we assume a patch is exhausted by taking one sample. If the agent chooses to continue searching it will be presented with a new object, drawn with probability \( P(X = x) \). The reward for taking a sampling action is information gained about the underlying distribution, which is the reduction in the entropy of \( P(Z|X = x) \) as a result of the latest observation. This reward function decreases in expectation, but in some instances may increase. However, if the distribution underlying the classes of objects changes, then we might expect to see the rewards increase for a period.

In this chapter we distributions underlying the classes of objects as Bernoulli random variables, representing whether a phenomenon of interest is present or not. In our grounding example of searching for life in a planetary science setting, it would model whether or not a material is colonized by microbes.

We place a Beta prior on the parameter \( \theta_x \sim Beta(\alpha_x, \beta_x) \) that determines the probability that a class of material is colonized. Post-observation we estimate that the probability of a class of object being colonized is \( \mathbb{E}[\theta_x] = \frac{\alpha_x}{\alpha_x + \beta_x} \). Here \( \alpha_x \) is the number of times material \( x \)
was observed being colonized ("success"), and $\beta_x$ is the number of times material $x$ was observed as not being colonized ("failure").

We anticipate that the agent will encounter a number $K = |X| \leq \infty$ classes of random variables while exploring. In different experiments we have different values of $K$. However the agent is never informed of how many classes of objects exist in the environment, and as such the algorithms must adapt as they encounter new classes of objects.

### 3.2.1 Algorithms

Three algorithms for sampling decision-making are evaluated in these experiments. Two of these algorithms estimate the value of action $x \in X$ by using Lindley’s value of an experiment ([Lindley, 1956](#)), given in Equation 3.1. This reward represents the expected information gain over all possible observations that may result from choosing to take sampling action $x$.

$$R(x) = H(\theta_x|z_{x,1:t}, x) - \mathbb{E}_Z [H(\theta_x|z_{x,1:t}, x)],$$ (3.1)

Where $z_{x,1:t}$ refers to the $t$ observations that were collected for object class $x$. The expectation over $Z$ is computed from the robot’s current belief in $P(Z|X = x)$, which is where the prior $Beta(\alpha, \beta)$ has influence. In our case we can compute this value exactly, because the space of possible values of $Z$ is small. However, for different random variables this might require estimating the value, if an closed-form solution is not available.

In the final experiment we consider a fourth algorithm, which is a modified form of the Foraging Algorithm. This algorithm attempts to detect whether the distribution underlying one of the classes of objects has changed. Should it do so, the algorithm caches the previous observations and re-initializes the algorithm.

**Greedy (Baseline Algorithm)** The first baseline algorithm, greedy sampling ([Algorithm 3.1](#)), will only choose to sample the encountered object, $x_t$, if it is has the highest reward compared to any other $x \in X$. This algorithm does not take into account the cost of moving to finding the next $x_t$, nor the rate at which they arrive. This algorithm corresponds to the simple greedy strategy of maximizing immediate reward.
Algorithm 3.1 Greedy Sampling Strategy

function INIT_GREEDY_SAMPLING
X ← ∅
R (·) ← ∅
end function

function GREEDY_SAMPLE(x_t)
if x_t ∉ X then
    X ← X ∪ x_t
    return ACTION_SAMPLE
end if
if R (x_t) > R(x) ∀ x ∈ X \ {x_t} then
    return ACTION_SAMPLE
else
    return ACTION_EXPLORE
end if
end function

Uniform (Baseline Algorithm) The second baseline algorithm, Uniform sampling, will choose to sample x_t if any other class of object, x ∈ X, has been sampled more than x_t. Like Greedy, this algorithm does not take into account the cost of traverse nor the cost of taking a sampling action. This algorithm attempts to distribute samples uniformly across all classes. Uniform sampling provides valuable comparison as it has been previously shown to be a robustly successful strategy Furlong and Wettergreen (2014b).

Algorithm 3.2 Uniform Sampling Strategy

function INIT_UNIFORM_SAMPLING
X ← ∅
N (·) ← ∅
end function

function UNIFORM_SAMPLE(x_t)
if x_t ∉ X then
    X ← X ∪ x_t
    N_{x_t} ← 1
    return ACTION_SAMPLE
end if
if ∃ x ∈ X \ {x_t} s.t. N_{x_t} < N_x then
    return ACTION_SAMPLE
else
    return ACTION_EXPLORE
end if
end function
**Foraging** The proposed algorithm, **foraging** (Algorithm 3.3), chooses to sample if the expected rate of reward of $x_t$ is greater than or equal to the expected reward from continuing to explore the environment and sample the next encountered object. We call the ratio of expected reward to costs the **productivity** of the algorithm. The foraging algorithm captures exploration and sampling costs, $J_e$ and $J_s$ in Algorithm 3.3 respectively, when making its decision.

We place a Dirichlet prior on the occurrence of the classes of objects, estimating the probability of encountering class $x_t$ as $\hat{P}(X = x_t) = \frac{n_{x_t}}{\sum_{x \in X} n_x}$, where $n_{x_t}$ is the number of times $x_t$ has been encountered. Initially, all classes of objects $n_x = 0$, so the algorithm only believes in the existence of a class of object after it has been observed. The distribution $\hat{P}(x_t)$ is used to compute the estimated value in the environment, in Algorithm 3.3.

**Algorithm 3.3 Foraging Sampling Strategy**

```plaintext
function INIT_FORAGE_SAMPLING
    X ← ∅
    R(·) ← ∅
    N ← ∅
end function

function FORAGE_SAMPLE(x_t)
    if $x_t \notin X$ then
        X ← X ∪ x_t
        $N_{x_t} ← 0$
        return ACTION_SAMPLE
    end if
    $N_{x_t} ← N_{x_t} + 1$
    sample ← $R(x_t) / J_s$
    explore ← $\mathbb{E}_X [R(x_t)] / (J_s + J_e)$
    if sample ≥ explore then
        return ACTION_SAMPLE
    else
        return ACTION_EXPLORE
    end if
end function
```

**Foraging with Change Detection** For the third experiment, we use a . The foraging with change detection algorithm uses the same decision rule as Algorithm 3.3 but after it makes an observation it checks to see if the distribution underlying those observations has changed, as in the function “DETECT_CHANGE” in Algorithm 3.4. It detects the change with a likelihood ratio test. It maintains two windows of observations for each $x \in X$, one which is initially populated with window size many observations, the other populated with the window size most recent observations.

The two observation windows represent hypotheses about the parameter $\theta_x$. A third window of the sample size most recent observations is used as the test population. We employ Wald’s sequential probability ratio test (Wald 1945) to determine if the observations in the second window
represents a different distribution from the first. We select the threshold for detecting a change in the distribution, \textit{change\_threshold}, as specified in [Wald, 1945]. \textit{window\_size} is arbitrarily set to be 30, and \textit{sample\_size} to 5. If a distribution change is detected the current world model is cached, and the rover resets its sampling algorithm to an initial state.

\textbf{Algorithm 3.4} Foraging with Change Detection

\begin{verbatim}
function INIT_FORAGE_SAMPLING
  X ← ∅
  \(R(\cdot)\) ← ∅
  \(N(\cdot)\) ← ∅
  \(\text{window}_a, x\) ← queue(∅)
  \(\text{window}_b, x\) ← queue(∅)
  \(\text{sample}_x\) ← queue(∅)
  \(\text{sample\_size} \leftarrow 5\)
  \(\text{window\_size} \leftarrow 30\)
end function

function DETECT_CHANGE(\(x, z_{x,t}\))
  if size(\(\text{window}_a, x\)) < \(\text{window\_size}\) then
    push(\(\text{window}_a, x, z_{x,t}\))
  end if
  push(\(\text{window}_b, x, z_{x,t}\))
  if size(\(\text{window}_b, x\)) > \(\text{window\_size}\) then
    pop(\(\text{window}_b, x\))
  end if
  push(\(\text{sample}_x, z_{x,t}\))
  if size(\(\text{sample}_x\)) > \(\text{sample\_size}\) then
    pop(\(\text{sample}_x\))
    \(\theta_{a,x} \leftarrow \text{sum}(\text{window}_a, x)/\text{size}(\text{window}_a, x)\)
    \(\theta_{b,x} \leftarrow \text{sum}(\text{window}_b, x)/\text{size}(\text{window}_b, x)\)
    \(\Lambda \leftarrow \sum_{j=0}^{\text{sample\_size}} \log \left( \frac{P(\text{sample}_x(j)|\theta_{a,x})}{P(\text{sample}_x(j)|\theta_{b,x})} \right)\)
  if \(\Lambda > \text{change\_threshold}\) then
    cache(\(\text{window}_a, x\)) \(\forall x \in X\)
    \(\text{window}_a, x \leftarrow \text{window}_b, x \ \forall x \in X\)
    \text{init\_forage\_sampling()}\)
  end if
end if
end function

3.3 Experiments

We conducted five experiments to demonstrate the effectiveness of our algorithm, varying the underlying distribution of each class, class arrival probability, and introducing a class distri-
bution change during the experiment. The costs of sampling and searching were varied over \( \{0.1, 0.2, 0.5, 0.75, 1.0, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \) for experiments 1 through 4. In all experiments, we ran 50 trials of each algorithm for each setting of experiment parameters and costs. In each experiment we consider different numbers of objects, \( K \), and we assume that the robot makes no errors in identifying these classes.

### 3.3.1 Experiment 1 - Uniform Arrival Distribution, Different Underlying Distributions

In the first experiment the arrival probability is fixed with a constant uniform distribution. That is to say the probability that the next random variable to be presented to the agent is \( P(X = x) = 1/3 \). In this experiment we vary the underlying distribution of the random variables, \( P(Z|X = x) \), which is the probability that objects in class \( x \in X \) is colonized.

#### Table 3.1: Experiment 1 Parameter Settings

| Experiment | \( P(Z|X = 1) \) | \( P(Z|X = 2) \) | \( P(Z|X = 3) \) |
|------------|-----------------|-----------------|-----------------|
| 1.1        | 0.01            | 0.50            | 0.99            |
| 1.2        | 0.01            | 0.30            | 0.01            |
| 1.3        | 0.01            | 0.50            | 0.01            |
| 1.4        | 0.01            | 0.75            | 0.01            |
| 1.5        | 0.01            | 0.99            | 0.01            |

### 3.3.2 Experiment 2 - Skewed Arrival Distribution with Identical Underlying Distributions

We consider arrival probabilities which are highly skewed such that there is one overwhelmingly available class of object. We use two kinds of distributions to examine this case. The first, with \( K = 6 \) objects, we set the arrival probabilities as per Table 3.2.

#### Table 3.2: The highly unbalanced arrival distributions used in this experiment. One class is given a majority of the probability mass, while the remaining probability mass is shared among the remaining classes of objects. In these experiments we let \( K = 6 \).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( P(X = 1) )</th>
<th>( P(X = x) ) ( \forall x \in {2, \ldots, K} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>0.9</td>
<td>0.1/(K-1)</td>
</tr>
<tr>
<td>2.2</td>
<td>0.8</td>
<td>0.2/(K-1)</td>
</tr>
<tr>
<td>2.3</td>
<td>0.7</td>
<td>0.3/(K-1)</td>
</tr>
<tr>
<td>2.4</td>
<td>0.6</td>
<td>0.4/(K-1)</td>
</tr>
<tr>
<td>2.5</td>
<td>0.5</td>
<td>0.5/(K-1)</td>
</tr>
</tbody>
</table>

We also consider a skewed distribution which is not as extreme. For this we use a Zipfian distribution \[ \text{[Newman, 2005]}, \] given in Equation 3.2. We consider different numbers of objects with \( K \in \{5, 6, 7, 8\} \). The list of experiments using this arrival distribution is given in Table 3.3.
\[ P_{\text{Zipf}}(X = x; s, K) = \frac{1/x^s}{\sum_{n=1}^{K} (1/n^s)} \]  

(3.2)

Table 3.3: Arrival distributions which follow Zipf’s law. These distributions favour some classes of objects much more than others. In these experiments we vary the number of classes of objects from \( K = 5 \) to \( K = 8 \).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( P(X = x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>( P_{\text{Zipf}}(X = x; s = 1, K = 8) )</td>
</tr>
<tr>
<td>2.7</td>
<td>( P_{\text{Zipf}}(X = x; s = 1, K = 7) )</td>
</tr>
<tr>
<td>2.8</td>
<td>( P_{\text{Zipf}}(X = x; s = 1, K = 6) )</td>
</tr>
<tr>
<td>2.9</td>
<td>( P_{\text{Zipf}}(X = x; s = 1, K = 5) )</td>
</tr>
</tbody>
</table>

In all experiments in this section we set \( P(Z|X = x) = 0.3 \ \forall \ x \in X \). We compare the performance of the Foraging algorithm against the Greedy and Uniform sampling algorithms.

3.3.3 Experiment 3 - Skewed Arrival Distribution with Distractor Object

In this experiment we consider Zipfian and skewed arrival distributions and we vary the underlying distribution of the second most common class of object, which for both distributions is \( X = 2 \). We have \( K = 8 \) objects in this experiment. The arrival distribution for all classes and the underlying distribution for the most common class is given in Table 3.4. \( P(Z|X = x) = 0.3 \ \forall \ x \in \{2, \ldots , K\} \). We want to determine how the effect of a object with a surprising underlying distribution will effect the behaviour of the Foraging algorithm.

Table 3.4: The Different settings for the arrival distributions of the different classes of objects and the distribution underlying the most commonly occurring class.

| Experiment | Arrival Distribution | \( P(Z|X = 2) \) |
|------------|----------------------|-----------------|
| 4.1        |                      | 0.1             |
| 4.2        | \( P(X = 1) = 0.8, P(X = 2, \ldots , K) = 0.2/(K-1) \) | 0.3             |
| 4.3        |                      | 0.5             |
| 4.4        |                      | 0.6             |
| 4.5        |                      | 0.1             |
| 4.6        | \( P(X = x) = P_{\text{Zipf}}(X = x; s = 1, K = 8) \) | 0.3             |
| 4.7        |                      | 0.5             |
| 4.8        |                      | 0.6             |
3.3.4 Experiment 4 - Skewed Arrival Distribution with Random Underlying Distributions

In the previous experiment the majority of the distributions underlying the classes of object were held constant at \( P(Z|X = x) = 0.3 \). In this experiment we consider four different settings of \( P(Z|X = x) \) which were drawn randomly from the interval \([0, 0.5]\). These parameters are given in Table 3.5.

Table 3.5: The underlying distributions used in experiment 5. For Experiments 5.1 to 5.4 we use the unbalanced distribution where \( P(Z|X = 1) = 0.8 \) and the remaining 7 objects share the remaining 0.2 of the probability mass. In experiments 5.5 to 5.8 the arrival distribution follows the Zipf distribution, \( P_{Zipf}(X; s = 1, K = 8) \).

| Condition | Experiments | \( P(Z|X = \{1, \ldots, K\}) \) |
|-----------|-------------|----------------------------------|
| RAND1     | 4.1,4.5     | \{0.14, 0.28, 0.28, 0.04, 0.27, 0.42, 0.38, 0.43\} |
| RAND2     | 4.2,4.6     | \{0.10, 0.32, 0.12, 0.32, 0.23, 0.39, 0.36, 0.06\} |
| RAND3     | 4.3,4.7     | \{0.04, 0.44, 0.01, 0.16, 0.31, 0.49, 0.03, 0.50\} |
| RAND4     | 4.4,4.8     | \{0.34, 0.33, 0.25, 0.07, 0.14, 0.39, 0.37, 0.31\} |

3.3.5 Experiment 5 - Distribution Change

In this experiment we compare the Foraging algorithm against the Foraging Algorithm with change detection, alongside a uniform sampling algorithm that collects samples for every sampling opportunity the foraging algorithm encounters. We compare only to a version of the Uniform sampling algorithm which already knows how many objects exist in the environment. Consequently it has an unfair advantage over the Foraging algorithm.

We return to \( K = 3 \) classes of objects, and fix the arrival probabilities \( P(X) = \{0.4, 0.3, 0.3\} \), and fix the sampling and searching costs at 0.01 and 0.1, respectively. Halfway through this experiment the underlying distribution is changed from \( P(Z|x) = \{0.001, 0.500, 0.001\} \) to \( P(Z|x) = \{0.001, 0.001, 0.300\} \). In the planetary setting this would represent moving between environmental regions, causing a change in colonization behaviour. The change occurs when the agent gets halfway along its path.

3.4 Results

To determine the success of an algorithm we measure the KL divergence between the true and estimated distributions. The performance of each algorithm is measured by the error in the algorithms’ estimate the underlying distributions by reporting the sum of the KL divergence of the underlying distributions for the different classes of objects. This performance metric is given in Equation 3.3.

\[
\text{error}(alg) = \sum_{x \in X} D_{KL}(\theta_x || \hat{\theta}_x) \tag{3.3}
\]
Figure 3.2: In this situation the Foraging algorithm performs as good or better than the Greedy algorithm. With the exception of the very high sampling and exploration costs in Experiment 1.4 and 1.5.

For experiments 1 through 4 we present 3D plots showing how the effect on the error in estimating the parameters $\theta_x$ from using the Foraging algorithm over the baseline algorithms as cost of sampling, $J_s$, and exploring, $J_e$, are changed. In addition, we present 2D plots showing where either the foraging algorithm performs better than the control algorithm, or the control algorithm performs better than the foraging algorithm, or when their performance is indistinguishable.

We report the effect size with Cohen’s $d$ (Cohen, 2013). Cohen’s $d$ is the ratio of the mean to the standard deviation of the difference between the trials. Values greater than 1.3 are considered to be very large, above 0.8 to be significant, and below 0.5 to be moderate, and below 0.2 to be insignificant. In the plots of Cohen’s $d$ where the behaviour of the algorithms is indistinguishable we set the value to zero.

In the figures plotted, dark blue means that the foraging algorithm had superior performance, and the red regions mean that baseline algorithms had superior performance. In the plots of effect size where the graph is white and set to zero that means there was not a statistically significant difference in performance, to a 95% confidence level. Reports of the performance of the three algorithms relative to a strategy which always samples is available in Appendix B.

3.4.1 Experiment 1 Results - Uniform Arrival Distribution, Different Underlying Distributions

In this experiment we had a small number of objects, and a uniform arrival distribution we can see that the Foraging algorithm performs as good or better than either the Greedy (Figure 3.2) or Uniform (Figure 3.3) sampling algorithms. The Greedy algorithm has a slight advantage when there are very high sampling and exploration costs for Experiment 1.4 and 1.5, there the Greedy algorithm has a slight, but statistically significant improvement over Foraging. Across the board, Foraging performs as good or better than Uniform sampling, mainly where the are small sampling costs but large exploration costs.
Figure 3.3: In this situation the Foraging algorithm is as good or better than the Uniform algorithm, for all sampling and exploration costs.

We can see here that when the arrival distribution is uniform and the number of classes are small, the Foraging algorithm is an acceptable algorithm to use. However, as can be seen in figs. B.1 to B.3, the approach of sampling every opportunity is a competitive approach, and outperforms all the algorithms for large sample and exploration costs.

3.4.2 Experiment 2 Results - Skewed Arrival Distribution, Identical Underlying Distributions

When the arrival distribution follows the unbalanced distribution, and the underlying distributions are uniform, the Foraging algorithm generally performs as well or better than the Greedy algorithm, as seen in Figure 3.4. There is a small region where the Greedy algorithm performs better than Foraging, and that seems to be at low exploration costs, and shifts rightward as the arrival distribution becomes less unbalanced.
Figure 3.4: Foraging vs Greedy Algorithm, unbalanced arrival distribution. As the arrival distribution approaches uniform, the performance of the Foraging algorithm approaches that of Greedy. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark read means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

In comparison to the Uniform algorithm, we can see that the Uniform algorithm is statistically significantly better than the Foraging algorithm for many sampling costs and small to moderate exploration costs. However, for larger exploration costs the Foraging algorithm performs better than the Uniform algorithm, although that gap decreases as the arrival distribution becomes less unbalanced. Also of note, as the algorithm becomes less unbalanced the region where the Uniform algorithm is better becomes more diffuse, and has a higher lower bound on sampling costs before Uniform is better than Foraging. For small sampling and exploration costs Foraging is significantly better than Uniform.
Figure 3.5: Foraging vs Uniform Algorithm, unbalanced arrival distribution. As the arrival distribution approaches uniform, the performance gap between Foraging and Uniform algorithm decreases. Uniform has an advantage for moderate to large sampling costs, whereas Foraging tends to have an advantage for moderate to large exploration costs. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark read means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

Comparing all the algorithms to an approach that samples every object it encounters, as illustrated in figs. B.4 to B.6, we can see that while the Foraging algorithm performs strictly as good as or better. The Greedy and Uniform algorithms perform better than the Foraging algorithm for some cost settings, as discussed above, but occasionally they perform worse than always sampling.

When the arrival distribution follows Zipf’s law, as in Figure 3.6, we see that the Foraging algorithm is as good or better than the Greedy algorithm, with the small exception of moderate sampling costs. Foraging has a statistically significant improvement for large sampling costs when the number of classes of objects are larger. However, as this number decreases, the gap between Foraging and Greedy decreases.
When compared to the Uniform algorithm, in Figure 3.7, we see that Foraging is as good or better than Uniform for moderate to small sampling costs, and for moderate to large exploration costs. When samples are expensive the Uniform algorithm generally performs better with large effect size, and this area tends to be more diffuse as the number of objects are reduced.
Figure 3.7: Foraging vs uniform Algorithm. As number of objects is decreased the Foraging algorithm loses its advantage for large exploration costs. However, the degree of advantage that the Uniform algorithm has decreases with fewer objects. Uniform sampling has superior performance with large effect size for small exploration costs and large sampling costs. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark red means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

Again, while the Foraging algorithm does not always perform better than the Greedy or Uniform algorithms, we can see that it does not perform worse than an algorithm which always collects samples, as in Figure B.9 Uniform (Figure B.7) and Greedy (Figure B.8) perform worse than always sampling for small sampling and exploration costs.

We can see that the relative performance of the algorithms is sensitive to the arrival distribution. The Foraging algorithm has greater performance with small sampling and exploration costs. Effectively, when there are many opportunities to sample, the Foraging algorithm takes advantage of them when Greedy and Uniform do not. Uniform, however, will only sample an object if it is not the most sampled object. Consequently when the sample costs are expensive and the exploration costs are low, the optimistic stance of Uniform provides it a distinct advantage.

3.4.3 Experiment 3 Results - Skewed Arrival Distribution with Distractor Object

In this experiment we attempted to determine the effect of a distractor object on the relative performance of the Foraging algorithm. We varied the underlying distribution of the second most frequently encountered object to vary from the initial prior belief.

When the arrival distribution was the unbalanced distribution, there was no major change to the performance relative to the Greedy algorithm, as seen in Figure 3.8. Generally the Foraging algorithm is as good or better than the Greedy algorithm for a wide variety of sampling and ex-
ploration costs. There is a small region where the Greedy algorithm has a statistically significant, with small effect size ($d < 0.5$), improvement over Foraging.

![Figure 3.8](image)

**Figure 3.8:** The Foraging algorithm vs Greedy, unbalanced arrival distribution. There is no substantial change in performance across the different settings of the underlying distribution. Greedy has a slight advantage for sampling costs of about 1.5 and exploration costs of about 1.

When compared to the Uniform algorithm, Foraging is as good or better for moderate to large exploration costs, as in Figure 3.9. However, for small exploration costs, the Uniform algorithm is statistically significantly better and with large effect size. There is a small improvement in Foraging at the most extreme value of the underlying distribution of the distractor object.

![Figure 3.9](image)

**Figure 3.9:** The Foraging algorithm vs Uniform, unbalanced arrival distribution. The Uniform sampling algorithm maintains the advantage for moderate to large sampling and small exploration costs. There gap in performance between Foraging and Uniform is slightly decreased for moderate exploration costs in the case $P(Z|X = 2) = 0.1$. 

51
When the arrival distribution follows Zipf’s law, we see that the gap in performance between Greedy and Foraging is decreased for large exploration costs. The advantage that Foraging enjoys for small exploration and sampling costs is expanded, as can be seen [Figure 3.10] Greedy gains a small, but statistically significant advantage for small exploration costs and moderate sampling costs.

\[ P(Z|X = 2) = 0.1, N=50 \quad P(Z|X = 2) = 0.3, N=50 \quad P(Z|X = 2) = 0.5, N=50 \quad P(Z|X = 2) = 0.6, N=50 \]

**Figure 3.10:** The Foraging algorithm vs Greedy, Zipfian arrival distribution. The Foraging algorithm either outperforms or is statistically indistinguishable from the Greedy algorithm, except for moderate sampling and small exploration costs. At the smallest settings of exploration and sampling costs the Foraging algorithm has a statistically significant improvement over Greedy.

Compared to the Uniform algorithm, following a Zipfian arrival distribution, the Foraging algorithm does not change significantly as a function of the distribution underlying the distractor object. The Foraging algorithm is as good or better than Uniform for moderate to large exploration costs, and again for small sampling and exploration costs.
Figure 3.11: The Foraging algorithm vs Uniform, Zipfian arrival distribution. For moderate to large sampling costs and small exploration costs the Uniform algorithm is statistically significantly better than the Foraging algorithm. Otherwise the Foraging algorithm is as good or better than the Uniform algorithm. There is no major change in performance as a function of the change in the underlying distribution of the second most common class of object, \( P(Z|X = 2) \). Again, Foraging has a large improvement over Uniform for small sampling and exploration costs.

For both arrival distributions the Foraging algorithm continues to be strictly as good or better than an approach which samples everything it encounters, seen in Figure B.12. While the magnitude of the increase may not be as great, we see that the range where the Foraging algorithm outperforms the always-sample approach is broader than in either Uniform (Figure B.10) or Greedy (Figure B.11).

The results from this experiment demonstrate that the relative performance of the Foraging algorithm isn’t overtly sensitive the underlying distribution of the second most commonly encountered object. What we can conclude from this is that the Foraging algorithm preferentially collects quantities of information instead of chasing after surprises. However, one should consider, if the encountered classes of objects do vary significantly from the prior, perhaps an explorer should spend more time investigating them. Since the Foraging and Greedy algorithms uses the same reward function, and the Uniform algorithm doesn’t consider the observations collected at all, none of the algorithms can be said to be evincing a reacting to surprising observations.

3.4.4 Experiment 4 Results - Skewed Arrival Distribution with Random Underlying Distributions

In the previous two experiments we kept the underlying distributions of most of the objects constant at \( P(Z|X = x) = 0.3 \forall x \in X \). In this experiment we compared the Foraging algorithm to the baselines where the underlying distributions were set as in Table 3.5. As we can

\[
P(Z|X = 2) = 0.1, N=50 \quad P(Z|X = 2) = 0.3, N=50 \quad P(Z|X = 2) = 0.5, N=50 \quad P(Z|X = 2) = 0.6, N=50
\]
see, the difference in performance of the Foraging algorithm compared to the baseline algorithms is highly sensitive to the underlying distributions of the classes of objects.

When compared to the Greedy algorithm, for either the unbalanced (Figure 3.12) or the Zipfian (Figure 3.14) arrival distributions, the Foraging algorithm reduces its advantages, for all but the smallest sampling and exploration costs. In the case of the RAND3 parameter setting, the Greedy algorithm actually performs as good or better than the Foraging algorithm for many sampling costs and for moderate exploration costs.

![Graphs showing performance comparison between Foraging and Greedy for different settings](image)

**Figure 3.12:** Foraging vs Greedy, unbalanced arrival distribution. The gap in performance between Foraging and Greedy closes, and in some settings Foraging loses its advantage for moderate exploration costs.

The Uniform algorithm demonstrate a substantial improvement in performance for the settings RAND3 and RAND4, with an increase in the region where it performs as good or better than Foraging, as seen in Figure 3.13. We also see much larger regions where the performance between the two algorithms is not distinguishable at a 95% confidence interval.
Figure 3.13: Foraging vs Uniform sampling, unbalanced arrival distribution. The region where Uniform performs better than Foraging becomes more diffuse for the settings of the underlying distributions used in RAND3 and RAND4.

We see similar behaviour for the Greedy algorithm when the classes of objects follow a Zipfian distribution (Figure 3.14), as we do when following the unbalanced distribution. However, the Foraging algorithm is better able to preserve it’s advantageous performance for small sampling costs, across all settings of exploration costs. In all but the RAND1 setting, the Greedy algorithm is as good or better than the Foraging algorithm for moderate to large sampling costs.

Figure 3.14: Foraging vs Greedy, Zipfian arrival distribution. The Foraging algorithm preserves the advantage for small sampling costs, but behaves as good or worse than the Greedy algorithm for moderate to large sampling costs.

Using a Zipfian arrival distribution and comparing against the Uniform algorithm we see in Figure 3.15 that the Foraging algorithm preserves its superiour performance for sampling costs up to 3, for small to moderate exploration costs, compared to uniform underlying distributions as in Experiment 2. This is true across all four settings of the underlying distributions. However, Foraging loses the advantage when the exploration cost are large.
What we can conclude from these experiments is that the Foraging algorithm is quite sensitive to the values of the underlying distribution. The main observation is that the performance of the Foraging algorithm is much more variable for large exploration costs, but it preserves its superior performance when the sampling and exploration costs are small, relative to the total budget.

Comparing to the algorithm which always samples, we observe the rare situation where the Foraging algorithm performs worse. Shown in Figure B.15 when the arrival distribution is unbalanced and the test condition is RAND3, the Foraging algorithm is worse than an approach that always samples for large exploration and sampling costs. Otherwise Foraging is strictly as good or better than always sampling. Again, Foraging avoids the penalty when the sampling costs are small that the Uniform (Figure B.13) and Greedy (Figure B.14) incur.

### 3.4.5 Experiment 5 Results - Underlying Distribution Change

Figure 3.16 shows that the foraging algorithm with change detection performs substantially better than not using it. The leftmost bars show the performance of the algorithms with and without change detection immediately before the change in the underlying distribution. By employing change detection ("After" in Figure 3.16) we see the error in estimating the underlying distribution is profoundly reduced. Notice also that the uniform sampling algorithm, while it performs better after the change detection than the standard foraging algorithm, it is still statistically significantly worse than the foraging with the change detection. Obviously, the uniform algorithm could also benefit the change detection mechanism.

If the number of opportunities to sample is not very large, then the agent will not be able to detect a change in an underlying distribution with any confidence. This must be considered when planning exploration missions. The number of samples that are needed to estimate the
distribution with confidence can be determined through PAC learning bounds, but would require a confidence level determined by the designers of the mission.

KL Divergence when Distribution Changes, N=50

![KL Divergence when Distribution Changes, N=50](image)

**Figure 3.16:** All algorithms perform comparably just before the underlying distribution change. At the end of the path the algorithm that detects changes performs substantially better than the ones that don’t. Error bars represent a 95% confidence interval, estimated from 50 trials.

The sampling/exploration cost that this experiment takes place in corresponds to the region identified in the previous two experiments, and hence we see that the uniform algorithm outperforms the foraging algorithm. However, the fact remains that by recognizing a change in the underlying distribution the algorithm is able to better estimate the underlying distribution.

### 3.5 Discussion

While improving the modelling of field work, this approach still has some limitations. For example, the probability of encountering different classes of objects is considered to be independently and identically distributed (i.i.d). While this may be a valid first-order approximation, it could extend the utility of the algorithm to consider a Markov chain with a limited time horizon to estimate the value of continued sampling.

We also consider fixed sampling and search costs. While this work focussed looked at different regimes of sampling and exploration costs, it could increase the fidelity of the simulation to introduce randomness into those costs.

The decision about whether or not to sample the currently available object is based on a hard comparison to the estimated value of exploring the environment. There are three underlying assumptions in this decision.
First, it is assumed that one bit of information gained is worth one unit of cost. It may be that a different relative weighting of rewards and costs would be more beneficial to behaviour. Since the decision being made by the robot is \( \frac{\text{sampling value}}{\text{sampling cost}} \geq \frac{\text{exploring value}}{\text{exploring cost} + \text{sampling cost}} \), when the cost of sampling is very small, and in particular much smaller than 1, it could ensure that the algorithm will never give up on a sampling opportunity. However, when both sampling and exploration are cheap, this is not unacceptable behaviour.

Second, it is assumed that the value of classes of objects is determined exclusively by the information gained. While this prevents against confirmation bias on the part of the robot, it does not permit scientists to impart preferences.

Third, it is assumed that a hard decision boundary between the value of current sampling opportunity and the estimated value of the environment is a suitable decision making mechanism. A softer boundary could reduce the propensity of the foraging algorithm to give up on sampling opportunities.

In the final experiment in this chapter, the foraging with change detection algorithm kept fixed the number of old and new samples that are compared, and the confidence threshold for detecting a change. These parameters can greatly impact the ability to detect changes, and furthermore could conceivably mask changes in the underlying distribution which occur slowly over time. While hard changes in the underlying distribution are anticipated in the situations which inspired this work, in other scenarios this assumption must be re-examined.

The sampling budget that the algorithms were tested with was fixed arbitrarily at 100 units. While the costs of sampling and exploring were varied, the total budget was not. We would expect that this would change the topography of the surfaces of the KL divergence and confidence bounds for the algorithms.

We also consider that the robot is able to identify the different classes of objects without error. Many template-matching algorithms deployed in situ effectively make this assumption as well. While this may be possible in some settings, when perception systems are deployed in unstructured environments this is almost certainly not true. Accounting for that confusion would make opportunistic sampling algorithms better suited to deployment in realistic systems.

This work can be extended in several ways. First, employ the same change detection of the sample values to the arrival probabilities. This way the exploring agent can detect when the composition of the environment changes, which may be interesting to remote scientists. Second, model more complex underlying distributions. Third, integrate site selection with a path planner in order to determine costs of different sampling actions. Finally, account for possible misclassification of the identified random variables in a scene. These additions will make progress towards robust autonomous planetary exploration.

The Foraging algorithm performs best relative to the Greedy and Uniform sampling algorithms when the sampling and exploration costs are small. Effectively, Foraging can recognize when it is sensible to engage with most of the encountered objects. However, it clearly loses out to the Uniform algorithm when the sampling costs are large and the exploration costs are small. While Foraging still out-performs a strategy of sampling everything for these cost settings, it would be good to enable Foraging to be more Uniform sampling in these conditions, without losing Foraging’s advantages when sampling costs are small.

It would be good to consider the relationship between environmental conditions and the results of sampling objects. A search procedure using one motivated by Quadratic Cauchy-
Schwartz Mutual Information would be an natural method of valuing these samples - how indepen-dent are environmental parameters and the distribution underlying classes of objects. However, this does require knowledge about environmental conditions, and once we have this information in a map we then get into the notion of planning. Further, it assumes that the relevant environmental parameters are observable.

Finally, it was noted that different points of the sample/exploration cost space yield different performance for the competing algorithms. It would be valuable to develop an algorithm that can estimate the relative costs, and then use that estimate to modify which sampling strategy they should employ. This should yield greater flexibility and performance of the algorithms overall.

### 3.6 Summary

In this chapter we examined the case of a robot following a pre-determined trajectory while attempting to learn the distribution underlying a number of classes of objects. The fundamental idea underlying the work in this chapter is the recognition that an agent does not necessarily have the choice of which class of object to sample. We proposed a foraging algorithm to address this shortcoming. Additionally we consider a modified version of the algorithm which can identify changes in the underlying distributions.

When the likelihood of encountering the different classes of objects is uniformly distributed we see that the Foraging algorithm generally performs better than the Uniform sampling algorithm. When the costs of sampling and exploring were high the Greedy algorithm had a slight performance advantage over the proposed Foraging algorithm.

When the distribution governing which classes of objects were encountered was non-uniform we saw this performance gap narrow, for both the Greedy sampling algorithm and the Uniform algorithm, with an increase in regions where the baseline algorithms outperform the Foraging algorithm. For larger settings of the sampling cost, the Uniform sampling algorithm out-performs the Foraging algorithm. However, the Foraging algorithm rarely under performs the Always Engage algorithm, as can be seen in Appendix B.

The result of the experimentation in this chapter demonstrate that there is not one clear winner between the Foraging, Greedy, and Uniform sampling. When exploration costs are small and sampling costs are large, the Uniform algorithm produces the best results. However, when sampling costs are small, the Foraging algorithm can yield improvements. There are also many settings of the parameters where the algorithms perform statistically indistinguishably from one another.

We must conclude that are different scenarios when different sampling strategies have the best performance. When the arrival distribution of the classes of objects is near uniform, or the number of objects is small, it makes sense to choose the Foraging algorithm, however, once that distribution becomes unbalanced, the Foraging algorithm begins to lose out for large sampling costs, motivating the choice of the Uniform algorithm. One can consider the costs for sampling and exploration, and then decide which mode of behaviour it would be most beneficial to employ.

However, if one must choose a single alternative to the approach of always engaging with samples, then we can say with confidence that the Foraging algorithm performs generally as good as or better than that strategy. Neither the Uniform or the Greedy algorithms can make this
claim for the tested search and exploration costs, and arrival distributions.

The Foraging algorithm demonstrates its greatest advantage when the sampling and exploration costs are small. It can recognize when it is sensible to engage with most of the encountered objects. While, it clearly loses out to the Uniform algorithm when the sampling costs are large and the exploration costs are small, Foraging still performs better than the strategy always sampling. The Foraging algorithm can be viewed as a compromise point between the Uniform sampling algorithm’s optimism, in the case of large sampling costs and small exploration costs, without losing the opportunistic zeal of a strategy that always chooses to sample.

An alternative view of this work is, if one has a secondary sensor where the cost of sampling can be controlled, and the mission has committed to a sampling strategy, then one can begin to reason about acceptable upper and lower bounds on that sampling cost. In this way the accuracy in individual samples can be traded against the overall accuracy in the estimates of the underlying distributions. This motivates a more rigorous theoretical investigation of the behaviour of Foraging algorithm, as without a theoretical relationship between the budget and the search and exploration costs this estimate could only at best be a heuristic.

We also demonstrated that an algorithm which monitored samples for changes in the underlying distribution was better able to estimate the underlying distributions after the change than algorithms which did not. Additionally, identification of these changes offers a trigger for autonomous explorers to examine the environment for changes causing the shifts in the underlying distribution. Regardless of the sampling algorithm used, change detection would be a valuable addition to any opportunistic sampling system.
Chapter 4

Opportunistic Sampling in a Scalar Fields (Prospecting)

Prospecting is a type of reactive sampling. One traverses a region with a sensor that is relatively cheap to use – the proxy sensor. The proxy sensor acts as a known indicator for some phenomenon to be investigated with an expensive, secondary, sensor. Prospecting is a distinct problem from the information foraging problem chiefly because it is not interacting with discrete targets, and because the relationship between the proxy sensor and the expensive sensor is already known.

During prospecting the agent is traversing a region with a cheap proxy sensor looking to identify places or regions to engage in sampling activities that we call Area of Interest Manoeuvres (AIMs). Examples of an AIM would be collecting soil samples with, e.g., a drill, or to drive a tight spiral pattern, interrupting the planned trajectory. Generally, the robot is navigating through a scalar or vector field, as opposed to a space littered with discrete objects, and it needs to find the best - per some objective function - location to deploy an action.

In this chapter we assume that a trajectory has been determined by some mechanism external to the agent. We can write that trajectory as $\tau(t)$ where $t \in [0, \infty)$. The prospecting sensor collects observations $z_t$ that are drawn from some underlying distribution $G(t) = G(\tau(t))$. Which is to say that the distribution driving the readings from the proxy sensor is determined by where the agent is along the trajectory.

We seek to improve upon state of the art algorithms by being aware of the hypothesis about change points in the underlying distribution. Specifically, one change point when the distribution changes from $G(t) = G_1$ to $G(t) = G_2$ at some change point, $t_{cp}$. We build an algorithm upon the Sequential Probability Ratio Test (SPRT), developed by [Wald (1945)], to determine with confidence that a change in the underlying distribution has occurred. We demonstrate that being aware of the change in the underlying distribution can produce superior performance to state of the art approaches used in the field.

What we call prospecting is different from sample scheduling in that it is reactive to the observations collected in situ along the trajectory. Further we do not place a constraint on how many discrete actions should be deployed, we leave that determination to a higher-level component in a robot system. As previously stated we assumes a pre-determined trajectory. While intelligently planning a trajectory could increase the performance of prospecting it does not eliminate the need
for a mechanism which makes opportunistic decisions to take discrete actions, so in this way the prospecting problem is different from planning for information gathering.

The work in this chapter was modelled on the Mojave Volatiles Prospector Project conducted in the Mojave desert in 2014 in part by the NASA Ames Intelligent Robotics Group. Here the prospecting instrument was a neutron spectrometer (NSS) and the high-cost intervention was an Area of Interest Mapping Manoeuvre (AIM). In this chapter we model an AIM as driving in a circular pattern when an area of interest is detected. The data used in this chapter was either data from the MVP project or simulated based on the distribution of NSS readings collected during this project. The MVP project is discussed in section 4.1.

The state of the art in prospecting is best exemplified by the work of Ferri et al. (2010). As such this algorithm is used as a baseline for comparison in the experiments conducted in this chapter. This algorithm issues an AIM if an observation crosses a predetermined threshold. This and other related work are discussed in section 4.2.

What we propose is an algorithm that attempts to determine if the source driving the readings of the prospecting sensor has changed. If a change has occurred then the robot can take any necessary actions. This algorithm is based around the sequential probability ratio test (SPRT) (Wald, 1945), which is used to determine if one of two hypotheses are more likely, given the observed data. The competing hypotheses are either that there has been no change in the underlying distribution or there has been one change in the underlying distribution.

The algorithms will be discussed in detail in subsection 4.3.1. The experiments used to test them will be described in subsection 4.3.3 and their results discussed in section 4.4.

4.1 The Mojave Volatiles Prospector Project

To ground this work we consider the Mojave Volatiles Prospector (MVP) project conducted at by NASA Ames in the Mojave desert in 2014 (Heldmann et al., 2015). The robot deployed in this project, pictured in Figure 4.1, was equipped with a Neutron Spectrometer (NSS), which was used to estimate the abundance of subsurface water.

MVP served as an early test of the high-tempo scientific operations that would be required to support short-duration exploration missions like the anticipated Resource Prospector mission (Andrews et al., 2014). Robot-habitable conditions on the surface of the moon, (e.g. sunlight, temperatures above freezing) last for only two week periods at a time. Unless a vehicle is designed to last lunar night, adding cost to the mission, operations must be limited to the lunar day. Consequently these missions do not have the luxury of Mars missions to take 24 hour periods to make decisions about deploying scientific instruments.

The Neutron Spectrometer (NSS) is an instrument that infers the abundance of water molecules in the field of view of the instrument. The instrument can sense water content up to a meter below the surface of the ground and in approximately a meter radius parallel to the ground plane.

The NSS reports counts of how many neutrons it estimates passed through the field of view of the instrument in one second blocks of time. These counts are modelled as a Poisson process. The average rate of the Poisson process is a function of the water abundance by weight and the depth of the water deposit. Generally speaking, a higher rate of counts implies a greater likelihood of water.
In the process of the MVP project it was discovered that the background rate was approximately 40 counts per second. A high number of counts, highly correlated with water is approximately 120 counts per second. A threshold for having detected water was approximately 80 counts per second. As such we used 80 as the threshold for the threshold algorithm used in this chapter.

The NSS is always on and does not consume resources other than power. Therefore we assume no sampling budget is consumed when using the NSS. We consider the fixed power budget of the NSS while prospecting as non-negotiable.

Prospecting is an important part of exploration, especially for missions focussed around in situ resource utilisation. An NSS is planned to be part of NASA’s upcoming Resource Prospector mission. The instrument will be the primary prospecting sensor in the search for subsurface deposits of solid water. The vehicle will drill at where the water density is most likely highest. The tight time constraints of operating in one lunar day motivate effectively finding likely water density maxima as observed by the neutron spectrometer.

4.2 Prior Work

There are three major non-adaptive methods for determining where and when to deploy AIMs: Delegate to a human; scout out the area and return to relative maxima; or to use a threshold to determine when an AIM should be employed. However, human judgement is not always available, or if it is, it requires demands be placed on unbiased experts whenever the prospecting sensor is deployed.

Pre-scouting an area is a viable approach should the exploration budget have sufficient mar-
gin that it can traverse the length of the path at least twice. But this approach does make the assumption that the interesting observations that are made along a traverse have no temporal component and will still be there when the robot returns to the site. This is not necessarily a valid assumption, and as such decisions that are made reactively would be more effective.

Yoerger et al. (2000) use a robot to scan an area by searching a grid. This approach is principled and avoids collecting data. However, erroneous readings can be over represented in the data, as there is no attention paid to the statistical confidence in the reported data. Likewise, an anomalies that do appear are uninvestigated, as this is a non-adaptive approach. The work presented in (Yoerger et al., 2007) improves on their prior approach by having the vehicle follow a search pattern, then identify anomalies – readings that pass a certain pre-determined threshold – and then return along the path to gather more observations of those anomalies.

The third approach, of using a threshold to determine when to conduct an AIM is, in principle, feasible, but only if the statistics of the environment are already well understood going into the exploration site. A static-threshold approach does not adapt to the statistics of an environment, and as such it may execute AIMS either too frequently, in sufficiently abundant territories, or not frequently enough, should sites with high concentrations of the reading of interest be missed, or the higher concentration regions be not as highly concentrated as the remote scientists hoped.

The paper (Ferri et al., 2010) describes a robot that is attempting to localize undersea hydrothermal vents. They track the vents by looking at the concentration of chemicals in the water which are emitted from hydro-thermal vents. Should the observed concentration pass a threshold they initiate a spiral area of interest mapping manoeuvre in order to build a more informed map of chemical concentrations.

In the case of the Resource Prospector project, the purpose is to localize maxima of the neutron signal in order to drill in a location likely to be most abundant with water. In the case of the work of Ferri et al. (2010) the objective was to localize geothermal vents, places likely to contain new and exciting ecosystems.

The work in citeferri2010novel is a method for deploying AIMS in an online fashion, improving on (Yoerger et al., 2007). In the previous work the vehicle had to travel to the end of the trajectory before it determines where to deploy the AIMS, where as (Ferri et al., 2010) the vehicle can determine to deploy AIMS as it is travelling. The AIMS that the algorithm deploys are spiral motions which collected more observations with the chemical sensor.

In (Ferri et al., 2010) they present an algorithm which uses an adaptive threshold, and compare it to a baseline of a fixed threshold algorithm. They employ their algorithms on pre-recorded data from transects collected with the ABE robot.

As the robot travels along the length of the transect they increase and decrease the threshold based on how many AIMS have been deployed vs how many AIMS it was suggested to be deployed. The suggested number of AIMS deployed is not a hard upper limit on how many can be deployed.

The threshold, $\gamma$, is updated when a “patch” is ended. A patch is a region of the transect where readings come close to triggering an AIM, but do not necessarily trigger an AIM. A schematic of the decision rule for updating the threshold is given in Algorithm 4.1.
Algorithm 4.1 The algorithm checks to see if the number of AIMs deployed is keeping pace, as a fraction of the suggested number of AIMs, with the fraction of the transect covered. The algorithm starts off with a threshold, $\gamma$ determined by a default threshold, specified by the scientists, which is denoted $\gamma_B$. There is also defined a minimum threshold, $\gamma_{\text{min}}$.

```plaintext
transect_frac ← \frac{\text{distance travelled}}{\text{transect length}}
K ← 2
if \text{aims deployed} \leq \text{suggested aims} then
    \gamma_{\text{low}} ← \text{median}\{\max(p) : \forall p \in \text{patches s.t.} \max(p) < \gamma\}
    \gamma' ← \max(\gamma_B \times (1 - \text{transect_frac}) + \gamma_{\text{low}} \times K \times \text{transect_frac}, \gamma_{\text{min}})
    \gamma ← \min(\gamma', \gamma_B)
else
    \gamma_{\text{up}} ← \text{median}\{\max(p) : \forall p \in \text{patches}\}
    \gamma' ← \max(\gamma_B \times (1 - \text{transect_frac}) + \gamma_{\text{up}} \times K \times \text{transect_frac}, \gamma_B)
end if
```

However, the actual fraction of the distance travelled was estimated by the number of track lines in the lawnmower pattern that have been completed. In effect the algorithm attempts to ensure that at least the number of suggested AIMs are deployed by the end of the transect. As it gets closer to the end of the transect the threshold will be lowered towards the minimum allowed threshold, $\gamma_{\text{min}}$. However, this approach is focused more on ensuring the AIMs are deployed, and less on ensuring that it has successfully detected changes in the distribution that is driving the sensors.

The reliance on an \textit{a priori} threshold, in the case of the static policy in \cite{Ferri2010} means that the vehicle cannot adapt itself to operations in environments that have not been previously characterized. \cite{Ferri2010} proposed their adaptive algorithm addresses this by modifying the threshold in response to the data observed.

Comparing their adaptive algorithm to the fixed algorithm they classified the AIMs they deployed into one of three categories: AIM with confirmed vent, AIM with likely nearby vent, and AIM with no vent. They also counted the number of vents that were missed by the algorithms, the performance of which is summarized here in Table 4.1 and found on p27 of \cite{Ferri2010}.

\textbf{Table 4.1}: The success of the AIMs deployed by the adaptive and fixed threshold algorithms. Also reported the number of missed vents. Numbers come from Table 2 on p27 of \cite{Ferri2010}.

<table>
<thead>
<tr>
<th></th>
<th>Adaptive Threshold</th>
<th>Fixed Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirmed Vents</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Likely Vents</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>No Vent</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Missed Vents</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

If we include all likely vents in the count of successful vent discoveries, we can fit beta
distributions to the precision and recall of the algorithms. We compute precision by dividing the number of true positive identifications of vents by the sum of the true positives and the false positives. We compute recall by dividing the number of true positive identifications by the sum of the true positive identifications and the missed vents.

From this analysis, summarized in Table 4.2, we find the degree of belief in the improvement does not meet a 95% confidence level, but is still quite large at 84% probability of improvement. From a “gambler’s ruin” perspective, one should overwhelmingly choose the adaptive threshold over the fixed. The effect size of the improvement from using the adaptive algorithm is large, with a Cohen’s $d = 0.99$.

Table 4.2: A Bayesian statistical analysis of the improvement due to the adaptive threshold.

<table>
<thead>
<tr>
<th></th>
<th>Adaptive</th>
<th>Fixed</th>
<th>$P$(Adaptive $&gt;$ Fixed)</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.61</td>
<td>0.56</td>
<td>84%</td>
<td>0.99</td>
</tr>
<tr>
<td>Recall</td>
<td>0.85</td>
<td>0.69</td>
<td>84%</td>
<td>0.99</td>
</tr>
</tbody>
</table>

It should be noted that the behaviour of the algorithm proposed by Ferri et al. (2010) is dependent on a number of parameters determined by scientists in the field. There are opportunities to fine tune this algorithm to be more or less aggressive in how it deploys AIMs, but it also requires knowledge of the environment in the form of the default threshold, the minimum threshold, and the suggested number of AIMs to be deployed.

The work developed by Girdhar focuses on processing images in situ. The images that are collected by the robot are subjected to topic analysis to create meaningful clusters of images. As the robot explores, it scores the perplexity of the scene, or in the case of Girdhar and Dudek (2016) sub-regions of the scene. Topic perplexity in this case is a measure of how well the image is described by any one of the topics already discovered in the image database. When the topic perplexity is high for an image, the vehicle slows down in order to collect more observations of the perplexing scene. These vehicles do not plan with respect to overall mission objectives or goals, but simply employ their resources greedily.

There is an analogy between topic perplexity and the entropy of an image. Girdhar et al. model images as Bag of Words (BoW) vectors. Their topic content can be viewed as the dot product between the topics BoW vectors and their individual image BoW descriptions. The product between the images and all the topic models produces a distribution over the topics. An image that has a high entropy distribution over the topic models represents an anomaly that is not adequately described by the code book developed from the previous observations and as such warrants further investigation. Additional observations get added into the database that are used for producing the topic models, and as such there is an adaptation to the topic model to describe the environment the robot is operating in.

In Thompson et al. (2013) the focus is on deploying samples when there is a limited sampling budget. Here the explorer is carrying an instrument that is sampling at a predetermined rate and the only thing the robot can do to collect more samples of a phenomenon of interest is to slow the vehicle down. Further, the vehicle is forbidden from back-tracking. The focus of the work was constructing a non-stationary Gaussian Process kernel function in order to have the vehicle...
adapt to anomalies on-line. That being said, generating two distributions to model the data, one stationary and one not, could be used as the competing hypotheses that are considered in this work, should one need to make a discrete decision.

In the case of using an instrument such as a neutron spectrometer, as in this chapter, there is no reason to believe that there will be an explicit limit on the number of samples that can be collected. However, there may be a limit on overall mission time and/or power resources that can be deployed at any given time. Additionally, unlike (Thompson et al., 2013), our algorithm needs to make a decision to deploy a discrete action, and not control a parameter in a continuum. Lee et al. (2018) engage in the automatic localization of gamma-ray sources using a single ground vehicle. The experimenters had the robot following one of three different paths, a straight line, a spiral, or a lawnmower path, and used a Compton gamma camera to localize the sources of radiation. The Compton camera is capable of reporting three-dimensional about incident gamma-rays, and thereby enable localization of sources much more effectively. The work hinges on having a good estimate of the 3D location of the robot and directional information from the Compton camera.

Unfortunately the Mojave Volatiles Prospector and Resource Prospector projects only have access to a point sensor in the form of the NSS. Having such an instrument for neutrons would be highly valuable, and would spur an interesting extension of this work - automatically detecting changes in the number of neutron sources with confidence.

Wilson and Williams (2017) present an approach for designing trajectories for surveying an environment to maximize the observations of values, in their case depth, within a certain range. The algorithm operates within an prescribed boundary, and uses a Gaussian process to model the depth observed by the robot. The robot then follows the gradient of the depth model until it intersects with the boundary. Then the robot follows the contour of the depth measurements until it has identified a closed region within the proscribed boundary, and then decomposes that region into polygons that are search using a lawnmower pattern.

The approach developed by Wilson and Williams (2017) uses the Gaussian process to reactively change directions in order to identify the region to be decomposed. After the decomposition is designed the robot then deploys the lawnmower pattern without further reactive planning. This approach could be used to design the trajectories that the robot in our algorithm is following, at the overall mission level.

An approach which employs Bayesian optimisation is presented by Marchant et al. (2014). They use the Bayesian optimisation approach to find the maxima in an unknown function over a two-dimensional space. They apply their method to static and time-varying functions. They have a robot travelling at a fixed speed, and with a fixed sampling rate.

They speed up the solution to their POMDP problem by using the Monte Carlo Tree Search Abramson (1987). The operations of the robot in question is talked about in terms of days, and it takes the algorithm approximately 8 days to learn the dynamic function, which itself has a period of 1 day.

The work reported in Marchant et al. (2014) was conducted in simulation. Morere et al. (2017) implement the BO-POMDP algorithm with a UAV mapping an indoor environment. Because of the increased efficiency of the MCTS algorithm they are able to re-plan quickly on a commodity computer. Motion plans consist of a fixed number of cubic splines, the robot collects samples while executing the trajectory. When the trajectory endpoint is reached a new trajectory
is planned, incorporating the most recent observations.

In the work of Morere et al. (2017) there is no intra-trajectory reaction to the samples collected. But as the capacity to re-plan increases, approaches like those described by Morere et al. (2017) will obviate the need for algorithms like the one proposed in this chapter, if only to trigger replanning. While the algorithm only considers planning trajectories and not deploying distinct sampling actions, this is something that could be formulated into a POMDP framework, and benefit from the efficient planned they employ.

Given that radiation tolerant hardware lags the performance of commodity hardware by at least a decade, there is still a need fast-executing reactive behaviour while the robot is executing plans designed off-board the robot. However, even today algorithms like those described in (Wilson and Williams, 2017) and (Marchant et al., 2014) can be immediately useful as mission-planning tools.

(Alcantarilla et al., 2016) present an approach for using convolutional neural networks (CNNs) to do change detections on images sequences captured at two different times. The approach relies on having hand-labelled masks for where change has occurred. They do not construct a notion of confidence for the change.

At first glance, situation addressed in (Ferri et al., 2010) is a perfect application for algorithms that solve the secretary problem. The purpose of the secretary problem is to give the decision maker some data on which to build a distribution, and only seeks to make one decision instead of taking multiple actions, and does not consider if the underlying distribution is changing.

The secretary problem is a problem setting from statistical literature (Ferguson, 1989) which was constructed to determine when was the optimal time to stop collecting samples. The principle set up was this: A person is trying to hire a secretary. They are presented with a sequence of \( N \) candidates to consider for the position. Each candidate has a knowable score, relative to all the other candidates which can be determined by interviewing the candidate. The interviewer then can either choose to hire the candidate or dismiss them and go on to the next candidate. Once a candidate has been dismissed they cannot be recalled. The problem is to determine when is the optimal time to stop interviewing candidates. It was determined that the optimal solution for this problem was to interview the first \( N/e \) candidates, where \( e \) is Euler’s constant, and reject each one in turn. The interviewing process stops either when the first candidate who scores as good as or better than the best candidate in the first \( N/e \) candidates, or take the last candidate. This ensures finding the best candidate 37% of the time.

In principle this decision rule makes sense: Determine the distribution of the scores of candidates, then find an outlier with a high score. This problem does line up nicely with automatic AIM execution, the sensor readings from our proxy sensor maps nicely to the score of the candidates. As the robot continues to move we are encountering the sensed phenomena much like the sequence of candidates.

Work like that described in (Das et al., 2015) applies what is called the submodular secretary problem to deploying a fixed sampling budget across a transect. The submodular secretary problem deploys \( k \) samples across a transect by dividing it up into \( k \) segments, and running a standard secretary algorithm in each segment. This approach does come with the assumption that the value of all the encountered sampling opportunities come from the same distribution.

Distributing samples uniformly across sub-regions of an environment is a principled approach. But due to the nature of secretary algorithms each sample must be deployed on its
region of the transect. This could result in ineffectual deployment of resources, should some regions of the environment have a substantially lower amount of the quantity of interest.

Since the objective of [Das et al. (2015)] is to learn the relationship between the encountered environmental characteristics, that is a reasonable assumption. This approach does not necessarily translate to our setting. We know the relationship between the proxy sensor readings and the quantity we are interested in sampling, and we understand that the underlying distribution is changing as we traverse the environment.

There are other reasons why the secretary problem setting is not an exact match for our proposed problem. First, it does not consider noise in the sensor. However, this objection could be overcome through the use of sensor filtering.

Second, the candidates are considered to be drawn IID from the same distribution. In our setting there can be strong spatial effects on the population from which sensor readings are derived. That is to say, near a location of a water deposit, the readings will come from a distribution with a higher mean than those from farther away from the water deposit.

Third, it assumes that there is no cost to continuing to the next candidate, and that candidate cannot be recalled. This is not the case for autonomous exploration, moving between locations takes time, and any traverse comes with risk associated with the execution of tasks in an unknown environment. While there are algorithms for secretary problems that permit recalling past opportunities, such as the work in [Rocha (1993)], but these do not address changes in the distribution.

The strong spatial dependence creates a need to not only determine how many distributions there are, but also to determine when the vehicle should be confident that it is operating in either one mode or the other. Just modelling the distribution alone is not sufficient, as simply executing AIMs when the probability of being in one regime is higher than the others could result in excessive deployment of AIMs and an unnecessary use of mission resources.

In summary, there are applications for robots that need to determine when to deploy expensive activities, namely AIMs. To date the algorithms deployed are either greedily ignoring resources, using arbitrary and non-adaptive thresholds, or simply control speed of a vehicle proportionally to perceived anomalies without considering if there is statistical confidence that an anomaly has been observed. This chapter proposes an adaptive algorithm that determines when to deploy resources in response to statistical confidence that there has been an event that warrants observation.

### 4.3 Method

We consider three algorithms for detecting changes in the distribution driving the observations in the proxy sensor, which are described in [subsection 4.3.1](#). There are four experiments that are used to test the algorithms. The first two experiments test the effect of the magnitude of the change and the delay in the onset of the change in the distribution driving the observations from the proxy sensor. The third experiment tests the algorithms on data collected as part of the MVP project. Finally, the final experiment simulates the performance of the algorithms in deploying AIMs while operating in a two-dimensional environment. We also compare the performance of the proposed algorithm to an MCMC sampling model, and find the execution time does not
warrant further investigation at this point. Otherwise, the results of the experiments are presented in section 4.4.

4.3.1 Algorithms

We used three algorithms in this experiment. The state of the art algorithm, as used in (Ferri et al., 2010) is to issue an AIM whenever a reading crosses a threshold. We modify this algorithm slightly to make for a fairer comparison to the proposed algorithm. The threshold algorithm with memory described in section 4.3.1 is the baseline algorithm that is employed in the experiments of this chapter.

We employ a second baseline algorithm, which detects a change if the probability of the current observation is not within the 95% high confidence interval of a Poisson distribution based on previous data. This algorithm permits relative changes in the signal to be detected, and should be able to identify changes that occurred below the threshold of the Memory Threshold algorithm.

Additionally, in subsection 4.3.2 we compare the proposed algorithm to a similar MCMC-based approach. Although performance is comparable to the proposed algorithm, and the implementation of the algorithm could be considered more intuitive than the proposed algorithm we find that the time to compute solutions is significantly higher and does not warrant that adoption without additional hardware speed-ups.

Detecting Changes with a Threshold

This algorithm compares every observation $z_t$ to a threshold, $\gamma$ and if $z_t > \gamma$, or conversely if $z_t < \gamma$ then a change has considered to have happened. This algorithm is memoryless and only considers one observation at a time, and each in isolation.

When the process being observed is well characterized this is a useful algorithm. However, the threshold needs to be tuned to every different environment that a robot is operating in. This makes the algorithm very brittle, but it remains a state of the art algorithm.

An analogous version of this algorithm is given a probability distribution, consider a change has occurred when a reading has a probability $P(z_t | \theta) < \delta$ then consider a change has occurred. For every such setting of $\delta$ there is a corresponding $\gamma$ to threshold the sensor values. Again, this algorithm has no memory, and once one starts trying to accumulate data in favour of a change, we get into the proposed algorithm, the Sequential Probability Ratio Test.
Algorithm 4.2 Threshold Change Detection. This algorithm reports that a change has occurred whenever the observed value, $z_t$, is greater than the threshold, $\gamma$. Because the algorithm has no memory, it simply reports that the change has occurred at the current time step.

```
function INIT_THRESHOLD_CHANGE_DETECTION(\gamma)
    \gamma \leftarrow \gamma
    \hat{t}_{cp} \leftarrow \emptyset
end function
function DETECT_CHANGE(z_t)
    if $z_t \geq \gamma$ then
        \hat{t}_{cp} \leftarrow t
    else
        \hat{t}_{cp} \leftarrow \emptyset
    end if
    return \hat{t}_{cp}
end function
```

This algorithm is in line with the approach taken by Ferri et al. (2010).

Detecting Changes with a Threshold and Memory (Baseline Algorithm)

With little additional effort it is possible to have a slightly smarter algorithm based on thresholding the observed data. We add additional memory such that the first time an observation crosses the threshold, $\gamma$, we mark the current time as the start of the change. The algorithm reports the change point as being the time of the first observation that has crossed the threshold until the observation goes below the threshold again. This approach is coded in Algorithm 4.3.

Memory of when a change first occurred helps stabilize the algorithm’s estimation of when a change has occurred. Without memory the algorithm will think that every time the threshold has been crossed is a new instance of a change in the underlying distribution.
Algorithm 4.3 Threshold Change Detection with Memory. This algorithm reports that a change has occurred at the first observed value, $z_t$, is greater than the threshold, $\gamma$. Once the observed value goes below the threshold the change period is reset. This prevents the algorithm for constantly reporting that a new change has occurred even though it has been consistently occurring for a number of time steps.

$$
\begin{align*}
\textbf{function} & \quad \text{INIT\_MEMORY\_THRESHOLD\_CHANGE\_DETECTION}(\gamma) \\
& \quad \gamma \leftarrow \gamma \\
& \quad \hat{t}_{cp} \leftarrow \emptyset \\
& \quad \text{in\_change} \leftarrow false \\
\textbf{end function}
\end{align*}
$$

$$
\begin{align*}
\textbf{function} & \quad \text{DETECT\_CHANGE}(z_t) \\
& \quad \text{if } z_t \geq \gamma \land \neg \text{in\_change} \text{ then} \\
& \quad \quad \hat{t}_{cp} \leftarrow t \\
& \quad \quad \text{in\_change} \leftarrow true \\
& \quad \text{end if} \\
& \quad \text{if } z_t < \gamma \text{ then} \\
& \quad \quad \text{in\_change} \leftarrow false \\
& \quad \quad \hat{t}_{cp} \leftarrow \emptyset \\
& \quad \text{end if} \\
& \quad \text{return } \hat{t}_{cp} \\
\textbf{end function}
\end{align*}
$$

Detecting Changes with Adaptive Threshold (Baseline Algorithm)

The approach in (Ferri et al., 2010) had two major aspect: First they recorded the anomalousness of observation, and used an adaptive threshold. With this algorithm we employ a threshold adaptation mechanism similar to, but not identical to, that employed by Ferri et al. (2010). We give our implementation of the algorithm in Algorithm 4.4.

An important part of the algorithm is the identification of patches. For our purposes a patch is created when counts cross over the lower bound on the threshold, $\gamma_{min}$. Observations over $\gamma_{min}$ are added to the patch as they are observed. If observed values drop below $\gamma_{min}$ for more than $N_{patch}$ contiguous observations then the patch is closed. Whenever a patch is closed the threshold is updated.
Algorithm 4.4 Adaptive Threshold Change Detection with Memory. This algorithm reports that a change has occurred at the first observed value, $z_t$, is greater than the threshold, $\gamma$. Once the observed value goes below the threshold the change period is reset. This prevents the algorithm for constantly reporting that a new change has occurred even though it has been consistently occurring for a number of time steps.

\begin{verbatim}
function INIT_ADAPTIVE_THRESHOLD_CHANGE_DETECTION($\gamma_B, \gamma_{min}, N_s$)
    $\gamma \leftarrow \gamma_B$
    $\gamma_B \leftarrow \gamma_B$
    $\gamma_{min} \leftarrow \gamma_{min}$
    $\hat{t}_{cp} \leftarrow \emptyset$
    in_change, in_patch $\leftarrow$ false
    patches $\leftarrow \emptyset$
    current_patch $\leftarrow \langle \cdot \rangle$
    patch_count, $N_{AIM}, N_s, N_{patch} \leftarrow 0$
end function

function DETECT_CHANGE($z_t, s$) \text{ \quad \texttt{▷ detect_change requires, s, \% of transect completed}}
    if $z_t \geq \gamma \land \neg \text{in_change}$ then
        $\hat{t}_{cp} \leftarrow t$
        in_change $\leftarrow$ true
        $N_{AIM} \leftarrow N_{AIM} + 1$
    end if
    if $z_t < \gamma$ then
        in_change $\leftarrow$ false
        $\hat{t}_{cp} \leftarrow \emptyset$
    end if
    if $z_t \geq \gamma_{min}$ then
        patch_count $\leftarrow 0$
        current_patch $\leftarrow$ current_patch $\cup \langle z \rangle$
        if $\neg \text{in_patch}$ then
            in_patch $\leftarrow$ true
        end if
    else
        patch_count $\leftarrow$ patch_count +1
        if in_patch $\land$ patch_count $\geq N_{patch}$ then
            in_patch $\leftarrow$ false
            patches $\leftarrow$ patches $\cup \{ \text{current_patch} \}$
            current_patch $\leftarrow \langle \cdot \rangle$
            update_threshold(s)
        end if
    end if
return $\hat{t}_{cp}$
end function
\end{verbatim}
The algorithm takes as parameters a default threshold, $\gamma_B$, a minimum threshold $\gamma_{\text{min}}$, and a suggested number of AIMs to be deployed $N_s$. The algorithm also keeps track how far along the transect has progressed, and if the number of AIMs deployed is not keeping pace with the fraction of the transect covered, the threshold is adjusted. The algorithm used for modifying the threshold, $\gamma$ is given in Algorithm 4.5.

**Algorithm 4.5** The algorithm compares the number of AIMs that have been deployed to the fraction of the transect that has been completed. If a smaller fraction of the suggested AIMs have been deployed than the fraction of the transect have been completed then the threshold is lowered to encourage a greater number of AIMs to be deployed. If a larger fraction of AIMs have been deployed than transect covered, then the threshold is raised to discourage future AIMs from being deployed.

```
function UPDATE_THRESHOLD(s)
    if $N_{\text{AIM}}/N_s \leq s$ then
        low_thr ← median (\{max(p) : \forall p \in \text{patches s.t. } \max(p) < \gamma\})
        $\gamma' \leftarrow \max (\gamma_B \times (1 - s) + \text{low_thr} \times s, \gamma_{\text{min}})$
    else
        raiseThr ← median (\{max(p) : \forall p \in \text{patches}\})
        $\gamma \leftarrow \max (\gamma_B \times (1 - s) + \text{raiseThr} \times s, \gamma_B)$
    end if
end function
```

In our experiments we set $\gamma_B$ to be the same value as $\gamma$ in the Memory Threshold algorithm, and keep the number of suggested AIMs, $N_s = 50$. The minimum threshold, $\gamma_{\text{min}}$ is set to 53 for experiments 1, 2, and 4, and set to 45 for experiment 3.

We do not employ the same techniques for identifying anomalous readings as that used by Ferri et al. (2010), consequently our algorithm doesn’t detect changes that occur below the minimum threshold, $\gamma_{\text{min}}$. However, as will be seen in section 4.4, we get similar improvements in performance from using the adaptive vs a fixed threshold as Ferri et al. (2010).

**Detecting Changes with Relative Changes (Baseline Algorithm)**

The threshold-based algorithms described above only detected changes if observations cross a pre-determined level. An obvious other approach is to examine relative changes of the readings, and if readings cross a boundary then consider that a change has occurred. In this case we take a Poisson distribution and detect a change if the probability of the new reading is outside of interval that contains $\delta$ percentage of the distribution. We set the confidence level to 95% for all experiments in this chapter. The implementation of the algorithm is given in Algorithm 4.6.
Algorithm 4.6 The relative change detection algorithm considers that the underlying distribution has changed. When a change has been detected the new reading is taken as the mean of the new distribution.

```plaintext
function INIT_RELATIVE_CHANGE_DETECTION(\( \delta \))
    \( \delta \leftarrow \delta \)
    \( \lambda \leftarrow \emptyset \)
    \( \hat{t}_{cp} \leftarrow \emptyset \)
end function

function DETECT_CHANGE(\( z_t \))
    if \( \lambda = \emptyset \) then
        \( \lambda \leftarrow z_t \)
    end if
    if \( z_t \notin \text{interval}(P(Z), \delta) \) then
        \( \lambda \leftarrow z_t \)
        \( \hat{t}_{cp} \leftarrow t \)
    end if
    return \( \hat{t}_{cp} \)
end function
```

Detecting Changes with the Sequential Probability Ratio Test (Proposed Algorithm)

The sequential probability ratio test considers two hypotheses, \( h_1 \) and \( h_2 \). Each hypothesis implies some probability distribution about observations that were collected. The sequential probability ratio test asks the question which distribution better explains observed data \( z_1, \ldots, z_t \). We determined the relative probability of the data given the hypotheses is:

\[
R_t = \frac{P(z_1, \ldots, z_t|h_2)}{P(z_1, \ldots, z_t|h_1)}
\]  

(4.1)

If at any given time \( R > 1 \) then \( h_2 \) is more likely. If \( R < 1 \) then \( h_1 \) is more likely. However, if observations are considered to be independently and identically distributed then we can write the following:

\[
R_t = \prod_{i=1}^{t} \frac{P(z_i|h_2)}{P(z_i|h_1)}
\]

(4.2)

And if we take the log of this quantity we find:

\[
\Lambda_t = \sum_{i=1}^{t} \left[ \log(P(z_i|h_2)) - \log(P(z_i|h_1)) \right]
\]

(4.3)

\[
\Lambda_t = \sum_{i=1}^{t} \log(P(z_i|h_2)) - \sum_{i=1}^{t} \log(P(z_i|h_1))
\]

(4.4)
The log form of the equations has the advantage of letting us work with sums, which are easier to keep a running total of than products. Since likelihoods can become very small very quickly it is more numerically stable to work with the log of the likelihood function.

Wald determined thresholds $\alpha$ and $\beta$ based on the acceptable false positive and false negative rates for the test. If $\Lambda_t > \alpha$ then $h_2$ is accepted and if $\Lambda_t < \beta$ then $h_1$ is accepted and the experiment stops. For specified levels of error probabilities the SPRT is the test with the minimum expected stopping time ([Wald], 1945).

In this chapter $h_1$ is the hypothesis that no change has occurred and $h_2$ is the belief that a change has occurred. We consider the hypothesis that a time has occurred at every time point between 1 and $t$, the current time. We select the most likely time change. If that potential change point crosses a threshold $\alpha$ then we consider that a change in the underlying distribution has occurred.

Fitting two distribution to a data set should always fit the data better than one distribution. The implication of this is that $R_t \geq 1$ and hence $\Lambda_t \geq 0$. So we can ignore checking if we become certain that there has no change. In fact, because we don’t do anything until confidence has built that there has been a change, no further action is required of the robot except to continue driving. The aim is only necessary when we are confident that a change has occurred.

One risk is that because $\Lambda_t \geq 0 \forall t \geq 0$ there is the risk that with enough observations the quantity will eventually cross the confidence threshold, even if no underlying change has occurred. To mitigate this risk we set $\alpha$ to be larger than prescribed by Wald. The task of determining the confidence level at which an AIM should be deployed is left to the people who designed the mission and be a function of their tolerance of risk and the expected duration of the entire transect.

There are a number of assumptions we make here in order to implement this algorithm. First, we assume the data from a given distribution are all IID and coming from Poisson distributions. We consider that in the case of no change in the underlying distribution that it was drawn from a Poisson distribution with rate $\theta_0$. In the case were there has been a change in the distribution we call the rate before the change $\theta_1$ and the rate after the change $\theta_2$.

Since we do not know $\theta_0$, $\theta_1$, or $\theta_2$, we estimate them by the maximum likelihood estimates from corresponding samples, $\hat{\theta}_0 = MLE(\langle z_0, \ldots, z_T \rangle)$, $\hat{\theta}_1 = MLE(\langle z_0, \ldots, z_t \rangle)$, and $\hat{\theta}_2 = MLE(\langle z_{t+1}, \ldots, z_T \rangle)$, respectively, where $T$ is the current maximum time step.

After an AIM has been conducted the window of samples gets reset to the samples after the change point, $t$, and the process continues on as before. The implementation of the algorithm is given in 4.7.
Algorithm 4.7 Sequential probability ratio test-based change detection. This algorithm considers all the data that has been collected up to the present time, $t$, and determines whether or not there has been a change in the underlying distribution.

```plaintext
function INIT_Sprt_change_detection(s)
    s ← s
    data ← ∅
end function

function likelihood(z_1, ..., z_n, λ)
    ▷ Computes log likelihood for a Poisson distribution
    return log (\prod_{k=1}^{n} P_{\text{poisson}}(z_k|λ))
end function

function DETECT_CHANGE(z_t)
    data ← data + ⟨z_t⟩
    l_0 ← poisson_likelihood
    likelihoods ← ∅
    for all $k \in \{1, ..., t\}$ do
        $\lambda_1 ← \text{mean}(z_1, ..., z_k)$
        $\lambda_2 ← \text{mean}(z_{k+1}, ..., z_t)$
        likelihoods[k] ← likelihood(z_1, ..., z_k, $\lambda_1$) + likelihood(z_{k+1}, ..., z_t, $\lambda_2$)
    end for
    $\hat{t}_{cp} ← \arg\max_{k \in \{1, ..., t\}} (\text{likelihoods}[k])$
    if likelihoods[change_point] − l_0 > s then
        data ← data [\hat{t}_{cp}, ..., |data|]
        return $\hat{t}_{cp}$
    else
        return ∅
    end if
end function
```

This algorithm simply detects the changes in the underlying distribution. It is up to other mission objectives to make value judgements about the meanings of those values. For instance, observations that contradict remote sensing data may warrant a different level of interaction than data that is variable, but consistent with what is observed from orbit.

We can achieve a 2× speed up in the algorithm by realizing that we do not need to recompute the log likelihood of the first $1, ..., t$ data points for each $t$. However here I express the naive implementation above, for ease of communication.

### 4.3.2 Execution Time Comparison of SPRT to MCMC-Bayesian Change Detection

A principled method to approach this problem would be to employ an MCMC sampling method to determine if there has been a change in the underlying distribution. Modern approaches would have one construct a probability distribution over the change point in the underlying distribution,
then feed the model data and use, e.g., an MCMC sampling algorithm to determine the probability that a change has occurred for each time step, for example as conducted in [Adams and MacKay (2007)]. The MCMC model will produce a distribution over change points. One can use that distribution to select the most likely change point and act upon it, or pass the whole distribution to another decision making algorithm. The setup for such a problem is given in as follows:

\[ C_t \sim \text{Poisson}(r_t); \]  
\[ r_t = \begin{cases} 
\text{late} & \text{if } s < t \\
\text{early} & \text{if } s \geq t
\end{cases} \]  
\[ s \sim U(1, t_{\text{max}}) \]  
\[ \text{early} \sim \text{exp}(1) \]  
\[ \text{late} \sim \text{exp}(1) \]

Where \( s \) is the switch point between the \text{early} and \text{late} arrival rates. The arrival rates are drawn from an exponential distribution. They, in turn, drive the Poisson distribution that produces the counts observed by the NSS. This model, along with the observed data up until the current time, \( t_{\text{max}} \), is estimated with an MCMC sampler.

To illustrate the relative performance of our proposed algorithm vs the Bayesian-MCMC approach in Figure 4.2 we report the time it takes to execute an update in response to the latest sample for transects of length 100 samples.

We executed the two algorithms on an Intel Core i7 CPU 860, which has eight cores and a clock speed of 2.8GHz, with 16 GB of RAM. The algorithms were both implemented in Python version 2.7 running in Ubuntu 16.04 LTS. The MCMC algorithm was implemented using the pymc3 library [Coyle (2016)].

The time to complete an update is substantially larger for the MCMC approach than the SPRT algorithm. The average time for the MCMC algorithm to process a sample was 61.1 seconds, the average time for SPRT to process a sample was 0.0719 seconds. The reduction in processing time was approximately 61 seconds, with a 95% High Density Interval (HDI) of [58, 62] seconds and effect size (Cohen’s \( d \)) of 16.8, which is an very large effect. Because of the increased time execution we exclude MCMC-based solutions.
Figure 4.2: Here we can see the average execution time for the different algorithms as a function of the number of observations collected. Error bars represent one 95% confidence interval based on $N = 30$ trials. The SPRT algorithm is on the order of 100x faster than the Bayesian algorithm.

We also considered the execution the two different algorithms their performed. We compare the performance of the two algorithms by looking at the error in identifying when the change in the underlying distribution occurred, $E_{cp}$. We define that error as:

$$E_{cp} = |\hat{t}_{cp} - t_{cp}|$$  \hspace{1cm} (4.10)

Where $\hat{t}_{cp}$ is the estimated time of the change and $t_{cp}$ is the actual time of the change. In the ten trials we performed the change all occurred at the halfway point, $t = 50$ and the underlying
distribution changed from $\lambda_1 = 40$ to $\lambda_2 = 80$. We looked at the error that occurs before the change in the underlying distribution ($t = 25$), immediately after the time change ($t = 52$), a short time after the time change ($t = 60$) and at the end of the transect ($t = 100$). The change point detection performance is shown in Figure 4.3. Notice that immediately after the change in the distribution has taken place, the variability of the MCMC algorithm is very large.

![Change Point Prediction Error for MCMC vs SPRT Algorithm](image)

**Figure 4.3:** Considering the performance of the two algorithms over 10 trials. The performance is measured as the error between the estimated time of change in the underlying distribution and the true change time. We consider their performance before the change in the underlying distribution ($t=25$), shortly after the change in the underlying distribution ($t=52$ and $t=60$) and at the end of the transect ($t=100$).

Table 4.3 shows the relative performance of the MCMC algorithm and the SPRT algorithms in detecting the change point in the underlying distribution. In all cases the SPRT and MCMC had statistically significantly different performance.

Something we observed is that after the change had occurred at $t = 50sec$ the MCMC approach reported the change happened at $t = 49sec$ while the SPRT algorithm reported that the change occurred at $t = 50sec$. This difference can be viewed as slightly different interpretation of how to report when the change happened, when is the last time step before the new data is...
observed or when is the first time step corresponding to the new data. So while the difference between the two algorithms is statistically significantly different, and with a substantial effect size it can be viewed as demonstrating comparable behaviour.

More telling is that before the change has occurred the SPRT algorithm is superior to the MCMC algorithm to correctly reporting that no change has occurred. This is a statistically significant difference with a profoundly large effect size. This alone could be taken as motivation for not using the MCMC approach.

Granted, the MCMC algorithm described here assumes that there is at least one change in the underlying distribution. Making it more complex would enable it to identify that there might be zero changes in the distribution in a scene. However, making the model more complex would not make it less computationally intensive.

Table 4.3: In all cases there is a statistically significant reduction in the error in detecting the change point. Before the time change has occurred the SPRT algorithm is better at detecting there has been no change in the underlying distribution than the MCMC approach. However for all the cases after the change has occurred there is an error of 1 time step, which is a negligible error. We can say that the SPRT and MCMC algorithms have indistinguishable performance after a change has occurred.

<table>
<thead>
<tr>
<th>T (sec)</th>
<th>$\mu_{MCMC} - \mu_{SPRT}$ (sec)</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>24</td>
<td>[24, 24]</td>
<td>$1.87 \times 10^3$</td>
</tr>
<tr>
<td>52</td>
<td>1</td>
<td>[0.97, 1.03]</td>
<td>48.2</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>[1, 1]</td>
<td>$1.88 \times 10^3$</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>[1, 1]</td>
<td>$1.86 \times 10^3$</td>
</tr>
</tbody>
</table>

Because the performance of the SPRT algorithm is indistinguishable from that of the Bayesian algorithm after the change, and superior to it before the change in the distribution, and because the time performance is substantially better, we choose to use the SPRT algorithm in favour of the Bayesian/MCMC approach.

4.3.3 Experiments

We test the SPRT algorithm against the Memory Threshold and Relative Change algorithms in four different experiments. All of the experiments revolve around detecting a change from one underlying distribution to another. All experiments consider a sensor with noisy readings that follow a Poisson distribution. The experiments compare the performance of the algorithms:

1. As a function of the magnitude of change in the underlying distribution rate.
2. As a function of the delay in the onset of the distribution change.
3. On real MVP data.
4. On a simulated 2D environment using data drawn from the observations from the MVP project.
In all the experiments a time step was one second. For the Memory Threshold algorithm a threshold of $\gamma = 80$ was used. For the Relative Change algorithm we used a confidence interval of $\delta = 0.95$.

**Experiment 1 - Effect of Magnitude of Change in the Underlying Distribution**

In the first experiment the agent is presented with sequential data. There is a change in the underlying distribution half way through the simulated transect. The initial rate for the Poisson distribution is $\theta_1 = 40$ counts per second. The second rate is given by $\theta_2 \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. We ran 50 trials of each setting of $\theta_2$. The duration of the transect was $T = 100$ time steps. Algorithm parameter settings are given in Table 4.4.

**Table 4.4:** The parameter settings for the algorithms used in this experiment. $\gamma_{\text{min}}$ was chose to be 53 because that is the upper end of a 95% confidence interval of a Poisson process with a rate of 40.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Threshold</td>
<td>$\gamma = 80$</td>
</tr>
<tr>
<td>Adaptive Threshold</td>
<td>$\gamma_B = 80$, $\gamma_{\text{min}} = 53$, $N_S = 50$</td>
</tr>
<tr>
<td>Relative Change</td>
<td>$\delta = 0.95$</td>
</tr>
<tr>
<td>SPRT</td>
<td>$s \in {2, 4, 8}$</td>
</tr>
</tbody>
</table>

**Experiment 2 - Effect of Delay in Onset of Change in the Underlying Distribution**

In experiment 2 we kept the two different rates constant with $\theta_1 = 40$ and $\theta_2 = 100$. The duration of the transect $T = 400$ time steps. The time step at which the distribution changed was varied over $t \in \{0.1T, 0.2T, 0.3T, 0.4T, 0.5T, 0.6T, 0.7T, 0.8T, 0.9T\}$. We ran 50 trials for each budget. The algorithm parameter settings are given in Table 4.5.

**Table 4.5:** The parameter settings for the algorithms used in this experiment. $\gamma_{\text{min}}$ was chose to be 53 because that is the upper end of a 95% confidence interval of a Poisson process with a rate of 40. In this experiment we down selected the SPRT confidence level settings to just $s = 8$. This was done in response to the performance from the previous experiments.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Threshold</td>
<td>$\gamma = 80$</td>
</tr>
<tr>
<td>Adaptive Threshold</td>
<td>$\gamma_B = 80$, $\gamma_{\text{min}} = 53$, $N_S = 50$</td>
</tr>
<tr>
<td>Relative Change</td>
<td>$\delta = 0.95$</td>
</tr>
<tr>
<td>SPRT</td>
<td>$s = 8$</td>
</tr>
</tbody>
</table>

**Experiment 3 - Performance on Real-World Data**

We tested the algorithms on real-world data collected as part of the MVP project. The objective of the test was to detect changes in the underlying distribution that were annotated by humans.
We tested on 9 days of data collected during October of 2014. Algorithm parameter settings are given in Table 4.6.

Table 4.6: Algorithm parameters were the same as in Experiment 2, except the lower threshold for the Adaptive algorithm was lowered to 50, because otherwise the algorithm did not identify changes in a majority of the transects.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Threshold</td>
<td>$\gamma = 80$</td>
</tr>
<tr>
<td>Adaptive Threshold</td>
<td>$\gamma_B = 80$, $\gamma_{min} = 50$, $N_S = 50$</td>
</tr>
<tr>
<td>Relative Change</td>
<td>$\delta = 0.95$</td>
</tr>
<tr>
<td>SPRT</td>
<td>$s = 8$</td>
</tr>
</tbody>
</table>

Experiment 4 - Effect on Performance in 2D Operations

Finally, to explore what effect the algorithm would have on performance in a mission scenario we simulated exploration by a rover in an environment modelled on data from the MVP project. We consider the objective to be the same as in Ferri et al. (2010) - to localize the local maxima of the underlying distribution. We considered a maxima to have been localized if the vehicle drove over that point.

We add another algorithm, No AIMs, simply follows the pre-defined trajectory. This was added to be able to gauge an absolute improvement of the baseline and proposed algorithms. The algorithm parameter settings are given in Table 4.7.

Table 4.7: Parameter settings were returned to the values used in experiment 2.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Threshold</td>
<td>$\gamma = 80$</td>
</tr>
<tr>
<td>Adaptive Threshold</td>
<td>$\gamma_B = 80$, $\gamma_{min} = 53$, $N_S = 50$</td>
</tr>
<tr>
<td>Relative Change</td>
<td>$\delta = 0.95$</td>
</tr>
<tr>
<td>SPRT</td>
<td>$s = 8$</td>
</tr>
</tbody>
</table>

We tested the performance of the algorithms on 30 randomly generated maps and compared them against an additional algorithm which simply followed a lawnmower pattern across the landscape. A map of the subsurface water density was produced by randomly placing 1000 samples in a $50m \times 50m$ map and the constructing a Voronoi map from those samples. The map was approximated as a $100 \times 100$ cell grid, where cell width was $0.5m$.

In this experiment we did not consider trafficability or obstacle avoidance and simply followed a lawnmower pattern that was $50m$ wide with $5m$ spacing between switchbacks. An example of the map with the path superimposed is shown in Figure 4.4. At any point in time, $t$, when the rover was at location $(x, y)$ the rate of the Poisson distribution from which observations were sampled was given by $\theta_t = M(x, y)$, where $M(x, y)$ was the value of the map at location $(x, y)$.  

83
Figure 4.4: Here we see a simulated map with the planned lawnmower path superimposed on it. The rate of the Poisson distribution that samples were drawn from was determined by the location of the vehicle in the map.
4.3.4 Performance Metrics

In experiments 1 and 2 we used three performance metrics. The first was the false positive rate. This is the number of time steps before the change in the underlying distribution that were reported as changes. The second metric was the false negative rate. This is the number of time steps after the change in the underlying distribution where no change was reported. The third metric was the mean error in the reported time change. This is the average difference the reported time of change and the actual time of change, for all time changes reported.

In experiment 3 we recorded true positives, false positives, and false negatives. We identified changes in the NSS signal manually, and recorded the ability of the competing algorithms to identify them. We recorded this data for each data set, corresponding to data collected on different days of the MVP field work.

In the fourth experiment we considered the number of local maxima in the map of the water map captured by the vehicle during navigation. This metric is inspired by Ferri et al. (2010) as the objective of the robot was to capture the local maxima of the chemical densities it was tracking. Ferri et al. (2010) placed an upper limit on the number of AIMs that the robot could deploy, which we do not. To remove the number of maxima that would have been observed regardless of how many AIMs were deployed, we subtracted the number of maxima observed by the No AIM algorithm from the performance of all the other algorithms. In order to determine how effectively the algorithms use the AIMs we consider the ratio of maxima observed to AIMs deployed.

4.4 Results

Below we present the results in the experiments described above. In the first three experiments we found that the proposed SPRT based algorithm outperforms the baseline algorithms. In the fourth experiment we find that the Relative Change algorithm successfully captures more maxima in the water map, but we find that it is less productive in its deployment of AIMs than the SPRT algorithm. We compare the performance of the algorithms using pooled Bayesian hypotheses tests for experiments 1-3, and a paired Bayesian hypothesis test in experiment 4. We set a confidence threshold of 95% probability for a different in performance. When statistical confidence is achieved we report the effect size of the improvement in performance.

In order to understand how the changes are reported by the algorithms we direct the reader’s attention to Figure 4.5. This illustration shows that the change point determined by the Memory Threshold and Relative Change algorithms are unstable. They continue to increase their estimate of the time change in response to noisy observations, and after the time change has occurred. The SPRT with a confidence level set at 2 erroneously detects changes before the actual change in the underlying distribution. The SPRT algorithm with the confidence level set to 8 does not report changes before the actual change has occurred, and after the change the estimate is stable. This behaviour is typical in all the experimental settings.
Figure 4.5: The detected change points vs time. Both the Memory Threshold and SPRT ($s = 8$) report no change before the change has occurred. After the change has occurred both SPRT algorithms have stable predictions of the change point. The Memory Threshold, Adaptive, and Relative Change algorithms have unstable predictions, with the Relative Change algorithm predicting changes before the change actually occurs, which the Memory and Adaptive Threshold algorithms do not.
4.4.1 Experiment 1 Results - Effect of Magnitude of Change in the Underlying Distribution

In Figure 4.6 we see that with the exception of where there is no change between the two arrival rates the SPRT algorithm detects the change with high accuracy. The Relative Change algorithm has approximately a zero false negative rate, this is because it reports changes with high frequency, regardless of whether or not a change has occurred. We see that the Adaptive algorithm has similar performance to the Memory Threshold algorithm, with a slight improvement over the fixed threshold algorithm.

When there is no change in the arrival rates the SPRT algorithm has a much higher false negative rate. This, however, is desirable, as if there is no change in the arrival rate then we don’t want the algorithm to detect a change. In this experiment that it was the SPRT algorithm with a parameter set to 8 that had the best false negative rate.

False Negative Rate vs Change in Distribution Rate, $N = 50$

![Graph showing False Negative Rate vs Change in Distribution Rate](image)

**Figure 4.6:** Showing the rate of false negatives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. A peak at $\lambda_2 - \lambda_1 = 0$ is expected because this represents no actual change in the distribution. Error bars represent a 95% confidence interval, $n = 50$.

If we look at Table C.1, Table C.2, and Table C.3 we can see that all of the SPRT algorithms perform statistically significantly different from the Memory Threshold algorithm.

In Figure 4.7 we see that the false positive rate is independent of the change in the distribution driving the sensor readings. It is clear to see that the Relative Change algorithm is performing significantly worse than any of the other algorithms. The SPRT algorithm with a confidence level of $s = 8$ is the best performing SPRT algorithm.
Figure 4.7: Showing the rate of false positives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. The SPRT algorithm’s performance is independent of the change in background arrival rate. This graph reveals that it is possible to control the false positive rate through the SPRT confidence parameter. Memory and Adaptive Threshold algorithms have identical performance. The Relative Change algorithm has an exceptionally large false positive rate. Error bars represent a 95% confidence interval, \( n = 50 \).

In Table C.4 shows that the SPRT algorithm with the confidence level set to 2 performs demonstrably worse than the Memory Threshold algorithm. When the SPRT confidence threshold is set to 4 (Table C.5) and 8 (Table C.6) we see that the false positive rate of the SPRT is indistinguishable from that of the Memory and Adaptive Threshold algorithms, and in turn, indistinguishable from 0.

In Figure 4.8 we see that the Memory Threshold algorithm does not correctly identify a change until the second distribution begins to approach the threshold. By virtue of having an adaptive threshold, the Adaptive algorithm improves on the Memory Threshold algorithm. The Adaptive algorithm was able to detect smaller changes in the driving rate than the Memory algorithm, and it had an improvement on the error in predicting the change. However, like the Memory Threshold algorithm, its estimation of the change point were not stable, post-distribution change.

The SPRT algorithm with a confidence level of 2 tends to make premature predictions about the change, hence the average negative error. SPRT with confidence levels of 4 and 8 both have a negligible prediction error. Again, with there being no difference between \( \lambda_1 \) and \( \lambda_2 \), an error.
rate of 50 is ideal. It means the algorithms are identifying the change point as being either at the start or the end of the transect, either being a valid statement. SPRT with a confidence level of $s = 8$ is the best performing algorithm.

The Relative Change Algorithm has a high and fairly consistent mean error in the estimated change point. This is a function of the high number of false positives and the unstable performance after the distribution change.

![Figure 4.8](image)

**Figure 4.8:** For every time step that the algorithm reports a change, the error between the reported time step and the true change point. Error bars represent a 95% confidence interval, $n = 50$.

As shown in Table C.7, Table C.8, and Table C.9, the SPRT algorithms maintain statistically significant difference from the Memory Threshold algorithm for all cases except for there being no difference in the underlying distribution when the SPRT confidence level is 8.

The conclusion we draw is that the SPRT algorithm with a confidence level of 8 overall has the best performance over the range of different changes in the magnitude of the background rate. When the change in the underlying distribution is small, it has a slightly higher false negative rate than SPRT with confidence level 4, but this is more than compensated for by the lower false positive rate, and the better performance when there is no change in the distribution. From here on out we exclude other settings of the SPRT confidence level other than $s = 8$. 

89
4.4.2 Experiment 2 Results - Effect of Delay of Change Onset

Based on the results from the previous experiment, we down select the SPRT algorithms to only using a confidence level of 8. The SPRT algorithm once again outperforms the threshold based algorithm. The rate of false positives is largely unaffected by the change point of the algorithm, whereas the Threshold algorithm has a statistically significantly higher rate, see Figure 4.9. The Relative Change algorithm has a substantially larger false positive rate than the Memory Threshold, Adaptive, or SPRT algorithms.

![False Positive Rate vs \( t_{cp}, N = 50 \)](image)

**Figure 4.9:** For every time step that the algorithm reports a change, the error between the reported time step and the true change point. As the change point increases the false positive rate of the Relative Change increases, asymptotically towards 97%. Memory Threshold and Adaptive algorithms had identical performance. Error bars represent a 95% confidence interval, \( n = 50 \).

The false negative rate of the SPRT algorithm is lower than that of the Memory Threshold and Adaptive algorithms, with statistical significance and almost entirely large effect sizes, as per Figure 4.10. The false negative rate of the SPRT algorithm does increase as a function of the time change of the distribution. Simply, if there is an overwhelming amount of data from the first distribution, it becomes more difficult to identify a change late in the observation scheme.
The Memory Threshold and Adaptive algorithms maintain an approximately constant performance as a function of change in the transition point. We expect an average false negative rate close to zero because the background rate is changed to be above the threshold of the Memory Threshold and Adaptive algorithms, the probability of dropping below the threshold is low. The Relative Change algorithm has a higher false negative rate, and the variability tends to increase with the increase in change onset.

Figure 4.10: For every time step that the algorithm reports a change, the error between the reported time step and the true change point. Error bars represent a 95% confidence interval, \( n = 50 \).

The mean error in the detected time change for the SPRT algorithm is very close to zero, as we see in [Figure 4.11]. The mean error of the Memory Threshold decreases linearly with the delay in the change, whereas the Relative Change follows a roughly parabolic curve.

The Adaptive algorithm has better performance than the Memory Threshold algorithm. This is consistent with the results found by Ferri et al. (2010). The Memory Threshold algorithm has unstable predictions post-distribution change, as does the Adaptive algorithm. However, because the Adaptive algorithm increases its threshold as more changes are detected, it will reduce the likelihood that a change will be identified. This makes the post-distribution change estimation more stable, hence the reduced error rate.
The reason for the linear behaviour of the Memory Threshold algorithm is that it isn’t really, at its heart, designed to detect changes in the underlying distribution, it is designed to detect anomalies. Because of this, whenever the readings rise above or drop below the threshold the algorithm determines that an event worth investigating has happened, and it has no confidence that this is the case.

As a result of this, at different points in the transect the algorithms will receive an observation at $t_1$ and report that a change has occurred, that $\hat{t}_{cp} = t_1$. Then at a later time, $t_2 > t_1$, the observation will go below the threshold, indicating that it has left the regime of interest but at $t_2 + dt$ the observation may cross the threshold again, causing the algorithm to report that another change in the distribution has occurred, that $\hat{t}_{cp} = t_2 + dt$. As observations keep dipping below the threshold, the more likely that the Memory Threshold algorithm is to report additional changes in the distribution.

The closer the new background rate is to the threshold, the more frequently this is going to happen. The result is that the estimate time change grows with the number of observations collected by the Memory Threshold algorithm. As the time change increases there are fewer opportunities to trigger the Memory Threshold algorithm, hence the reduction in error.

Similar reasoning applies to the Relative Change algorithm. However, where the Memory Threshold algorithm’s estimate of the change point is only unstable after the change has occurred, the Relative Change algorithm has unstable estimates before and after the change, due to its high false positive rate. This error reaches a minimum when the change point is in the middle of the transect. The SPRT algorithm, however, maintains a stable estimate of the time change once it has detected the change.
Figure 4.11: The average error in predicting the time point of the distribution change. The SPRT algorithm has a very small error in predicting the time change, independent of the change point. The threshold algorithm, however, has a linear relationship in the error in predicting the change point. Relative Change algorithm has a roughly quadratic relationship, and never gets below 100 time steps error in estimating the change point. Error bars represent a 95% confidence interval, \( n = 50 \).

We see this in Figure 4.11. The average error of the change estimate, whenever a change is reported, is considerably higher for the Memory Threshold and Relative Change algorithms than the SPRT algorithm. The Adaptive algorithm has better performance than either the Memory Threshold and Relative Change algorithms, but still inferior to the SPRT algorithm. The error in the SPRT algorithm remains effectively constant, and small. Table C.12 shows that at all times the error in prediction for the SPRT was statistically significantly different from the Memory Threshold algorithm.

### 4.4.3 Experiment 3 Results - Real MVP Data

We tested the competing algorithms on data collected during the 2014 MVP operations in the Mojave desert. We are using the NSS data that was collected from the 17th to the 25th of October, 2014.

In this experiment we consider only counts from the stainless steel channel. Figure 4.12 to Figure 4.20 shows the data that were collected during the MVP project. We set the threshold and memory threshold algorithms with the same threshold that was used to identify hot spots in the
field. The SPRT algorithm was run with a confidence level of 8, the Memory Threshold was run with a threshold of 80 counts, and the Relative Change algorithm used a threshold of 0.95.

**Figure 4.12:** The blue line indicates the NSS counts recorded on 17 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.13: The blue line indicates the NSS counts recorded on 18 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.14: The blue line indicates the NSS counts recorded on 19 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.15: The blue line indicates the NSS counts recorded on 20 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.16: The blue line indicates the NSS counts recorded on 21 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.17: The blue line indicates the NSS counts recorded on 22 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.18: The blue line indicates the NSS counts recorded on 23 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.19: The blue line indicates the NSS counts recorded on 24 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Figure 4.20: The blue line indicates the NSS counts recorded on 25 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.
Table 4.8: In this table we report the results of the Memory Threshold, Adaptive, Relative Change, and SPRT algorithms on real-world data. While SPRT does report some false positives, they are rare. We also see that SPRT notices changes that are not observed by the Memory Threshold algorithm simply because they are happening below the threshold. This is done without adapting the algorithm to the environment.

<table>
<thead>
<tr>
<th>Date</th>
<th>Memory Threshold</th>
<th>Relative</th>
<th>Adaptive</th>
<th>SPRT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP</td>
<td>FP</td>
<td>FN</td>
<td>TP</td>
</tr>
<tr>
<td>2014-10-17</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>2014-10-18</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-19</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>2014-10-20</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>2014-10-21</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>2014-10-22</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-23</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-24</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>2014-10-25</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>2</td>
<td>0</td>
<td>90</td>
<td>39</td>
</tr>
</tbody>
</table>

We can use these aggregate data to fit Beta distributions over the precision, computed $\frac{TP}{TP+FP}$, and recall, computed $\frac{TP}{TP+FN}$, of the different algorithms. We can also use these distributions to determine the probability that any one algorithm’s performance is higher than the others.

Table 4.9: We estimate the precision and recall of the algorithms estimated from the data in the final row of Table 4.8. The Memory Threshold algorithm has a high precision, but the overall level of detection is quite poor, as seen in its recall score.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRT ($s = 8$)</td>
<td>0.94</td>
<td>0.74</td>
</tr>
<tr>
<td>Memory Threshold ($\gamma = 80$)</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td>Adaptive ($\gamma_{min} = 50$)</td>
<td>0.88</td>
<td>0.08</td>
</tr>
<tr>
<td>Relative ($\delta = 0.95$)</td>
<td>0.74</td>
<td>0.41</td>
</tr>
</tbody>
</table>

The precision and recall of the algorithms are given in Table 4.9 and the statistical significance of the relative performance is given in Table 4.10. As we can see the SPRT algorithm has a statistically significant improvement over the baseline algorithms except for Precision for the Memory Threshold algorithm. While the Memory Threshold algorithm does have superior precision, we can see that its rate of detecting changes is sufficiently low that the improvement in precision doesn’t warrant adopting the algorithm.
Table 4.10: The comparison of the SPRT algorithm to the Relative and Memory Threshold algorithms. We see that the SPRT algorithm has superior recall compared to either algorithm, with probability \( p \geq 95\% \), and huge effect sizes. The SPRT algorithm also has statistically significantly larger precision than the Relative change algorithm with probability \( p \geq 95\% \) and huge effect size.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Metric</th>
<th>( p \geq 95% )</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRT &gt; MemThreshold</td>
<td>Precision</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>SPRT &gt; MemThreshold</td>
<td>Recall</td>
<td>Y</td>
<td>14.83</td>
</tr>
<tr>
<td>SPRT &gt; Relative</td>
<td>Precision</td>
<td>Y</td>
<td>3.15</td>
</tr>
<tr>
<td>SPRT &gt; Relative</td>
<td>Recall</td>
<td>Y</td>
<td>4.92</td>
</tr>
<tr>
<td>SPRT &gt; Adaptive</td>
<td>Precision</td>
<td>Y</td>
<td>22.58</td>
</tr>
<tr>
<td>SPRT &gt; Adaptive</td>
<td>Recall</td>
<td>Y</td>
<td>12.51</td>
</tr>
</tbody>
</table>

4.4.4 Experiment 4 - Effect on Performance in 2D Operations

In Ferri et al. (2010) the objective was to localize maxima in the map. To determine the effectiveness of the SPRT algorithm we have simulated an experiment where a rover is following a lawnmower pattern across a terrain that has pockets of water density that will drive different average rates of the Poisson process for a NSS sensor.

The algorithm does not know that it is operating in a two dimensional environment. This choice was made in order to gauge the performance of the algorithm as trajectory-agnostic component to a robot scientist. Further, this mirrors the deployment of the threshold algorithm employed by Ferri et al. (2010). There are higher-level concerns that are ignored by the algorithm which can be delegated to a higher-level module of the robot, much like the CASPER system in OASIS/AEGIS.

We took data from the NSS readings from the MVP field experiment and fit a distribution over the number of counts. We sampled 100 observations from this distribution and used them as seeds to generate a Voronoi map, called \( M \), representing a 50m by 50m area, with a grid size of 0.5m.

The map was then smoothed with a Gaussian blur with a scale factor of \( \sigma = 0.8 \) in order to mimic the effect of the field of view of the sensor. At each point \( (x, y) \) in \( M \) the robot is given a sample from a Poisson distribution with \( \lambda = M(x, y) \). We generated 30 maps and ran ten trials on each map for each algorithm.

The robots follow a trajectory that follows a lawnmower pattern across the width of the map and with the swaths of the path having a spacing of 5m. The trajectory is illustrated in Figure 4.4. We compare three algorithms. The first simply follows the lawnmower path without conducting any AIMs. The second algorithm conducts an AIM when an observation crosses a threshold, as per Ferri et al. (2010). The third algorithm uses the SPRT algorithm described above, and when a transition is detected going from a lower background rate to a higher background rate it conducts an AIM, in the hopes of being on a gradient towards a local maxima.

Because the objective of the experiment in Ferri et al. (2010) was to localize the source of chemicals we set the robot in the simulation the task of collecting high-value observations. High value observations were defined to be local maxima in the water map. To score the performance...
of the robot we counted the number of instantaneous observations that were above the threshold over the course of the transect. To score the performance on localizing maxima we count the number of maxima that fell within the robot’s field of view while driving. Higher scores are better. A threshold of 80 counts was established in conversation with member of the MVP team.

To identify the number of maxima that are captured by virtue of using the different algorithms we first subtracted the number of peaks that were captured by not doing any AIMs. Figure 4.21 plots the number of unique maxima captured vs the number of AIMs that were deployed by the algorithms. The raw data is available in Table C.14 and Table C.13.
Figure 4.21: The number of maxima observed by the algorithms versus the number of AIMs deployed for the 30 different simulated maps. The Relative Change algorithm captured a greater number of maxima than the SPRT algorithm, and the SPRT captured a greater number of maxima than the Memory Threshold algorithm. The Adaptive algorithm deployed slightly more AIMs than the SPRT algorithm, and observed a similar number of maxima. The dashed line has a slope of 1. The further below the line, the less efficient the algorithm is in deploying AIMs. Each point represents $n = 10$ trials on each map.

The Relative Change, SPRT, and Memory Threshold algorithms were all statistically significantly different from one another in terms of both the number of AIMs deployed and the number of maxima captured. The Adaptive algorithm deployed a significantly different number of AIMs from the other algorithms, but the number of maxima captured was statistically indistinguishable from the SPRT algorithm at a confidence level of 95%.

The number of AIMs deployed and maxima observed by the algorithms are given in Table 4.11. The statistical significance of the differences in performance is given in Table 4.12.
We ordered the comparisons by decreasing number of AIMs deployed. With the exception of the number of maxima observed by the SPRT and Adaptive algorithms, all other quantities were distinct at a confidence level of 95%.

Table 4.11: The number of AIMs deployed and maxima observed by the different algorithms averaged over the 30 simulated maps. Relative Change deploys the most AIMs and observes the most maxima, followed by the Adaptive and SPRT algorithms, then the Memory Threshold algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AIMs Deployed (µ ± σ)</th>
<th>Maxima Observed (mu ± σ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Change (δ = 0.95)</td>
<td>108.5 ± 0.7</td>
<td>76.8 ± 4.7</td>
</tr>
<tr>
<td>Adaptive (γmin = 53)</td>
<td>57.7 ± 7.5</td>
<td>32.4 ± 3.14</td>
</tr>
<tr>
<td>SPRT (s = 8)</td>
<td>41.4 ± 2.2</td>
<td>35.8 ± 3.3</td>
</tr>
<tr>
<td>Memory Threshold (γ = 80)</td>
<td>2.2 ± 1.5</td>
<td>1.7 ± 1.6</td>
</tr>
</tbody>
</table>

Table 4.12: Relative Change deploys more AIMs than SPRT, and it observes more maxima than SPRT with an extremely large effect size at a confidence level of 95%. Similarly the SPRT algorithm deploys more AIMs than the Memory Threshold algorithm, and it also observes more maxima in the map, at a confidence level of 95% and a very large effect size. The number of maxima observed by the Adaptive threshold is statistically indistinguishable from the SPRT algorithm, while deploying more AIMs.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>AIMs Deployed p ≥ 95%</th>
<th>Cohen’s d</th>
<th>Maxima Observed p ≥ 95%</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Change &gt; Adaptive</td>
<td>Y</td>
<td>6.83</td>
<td>Y</td>
<td>10.4</td>
</tr>
<tr>
<td>Adaptive &gt; SPRT</td>
<td>Y</td>
<td>2.33</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>SPRT &gt; Memory Threshold</td>
<td>Y</td>
<td>14.79</td>
<td>Y</td>
<td>8.86</td>
</tr>
</tbody>
</table>

However, if we look at how effective the AIM deployments are, the picture changes somewhat. We can measure the effectiveness, of the algorithms by the ratio of maxima captured to AIMs deployed. If every AIM deployed resulted in discovering a unique maxima in the field, then the ratio would be 1. Values greater than 1 would be preferred, and the further below the dashed line in Figure 4.21, and the less effective the algorithm is.

The Memory Threshold algorithm and the SPRT algorithm are statistically indistinguishable in their relative effectiveness at a confidence level of 95%. The uncertainty for the Memory Threshold algorithm is much higher than the other algorithms because there were two trials where the Memory Threshold algorithm did not deploy any AIMs, and so they were excluded from the analysis. The effectiveness of the different algorithms is reported in Table 4.13.

The Adaptive algorithm had the worst effectiveness. We conjecture that is because, as the transect progresses, the threshold for deploying an AIM is reduced, should AIMs have not been deployed. If the threshold is lowered then AIMs are likely to be deployed in less favourable environments.
Table 4.13: The ratio of maxima observed to AIMs deployed. An effectiveness of 1 would mean that every AIM deployed observes a unique maxima in the underlying map.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Effectiveness $\left(\mu \pm \sigma\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRT ($s = 8$)</td>
<td>0.86 $\pm$ 0.07</td>
</tr>
<tr>
<td>Memory Threshold ($\gamma = 80$)</td>
<td>0.77 $\pm$ 0.45</td>
</tr>
<tr>
<td>Relative Change ($\delta = 0.95$)</td>
<td>0.70 $\pm$ 0.04</td>
</tr>
<tr>
<td>Adaptive ($\gamma_{\text{min}} = 53$)</td>
<td>0.57 $\pm$ 0.07</td>
</tr>
</tbody>
</table>

Using a paired significance test the SPRT algorithm has a higher effectiveness than the Relative Change algorithm at a confidence level of 95% and an effect size of $d = 2.56$. The SPRT algorithm has a greater effectiveness than the Adaptive algorithm at a confidence level of 95% and effect size of $d = 3.80$. The average different in effectiveness between the algorithms is given in Table 4.14.

Table 4.14: The difference in effectiveness for the different algorithms. At a 95% confidence level it is not possible to distinguish the performance of the Memory Threshold from the SPRT algorithm. There are two transects where the Memory Threshold algorithm did not deploy any AIMs, omitting these transects makes the performance of the Memory Threshold algorithm sufficiently variable as to make its performance indistinguishable from the other algorithms. The SPRT algorithm has superior effectiveness than either the Relative Change or Adaptive algorithms.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>$\Delta$ Effectiveness $\left(\mu \pm \sigma\right)$</th>
<th>$p \geq 95%$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRT $&gt;$ Relative Change</td>
<td>0.16 $\pm$ 0.06</td>
<td>Y</td>
<td>2.56</td>
</tr>
<tr>
<td>SPRT $&gt;$ Adaptive</td>
<td>0.30 $\pm$ 0.08</td>
<td>Y</td>
<td>3.80</td>
</tr>
<tr>
<td>SPRT $&gt;$ Memory Threshold</td>
<td>0.09 $\pm$ 0.46</td>
<td>N</td>
<td>N/A</td>
</tr>
</tbody>
</table>

What we can conclude from these results is that the SPRT algorithm is as efficient as the Memory Threshold algorithm, at a confidence level of 95%, but it collects a greater number of maxima in the underlying map. The Relative Change algorithm captures more maxima, but comes at the cost of significantly reduced efficiency, this is the result of the false positives observed in subsection 4.4.1 and subsection 4.4.2.

We see that the Adaptive threshold was able to capture more maxima than the Memory Threshold algorithm, which is consistent with the results reported by Ferri et al. (2010). However, with our formulation of the adaptive threshold, we see that the adaptive algorithm is less productive, largely because of how it lowers its standards for deploying an AIM as the threshold is lowered.
4.4.5 Discussion

The algorithms presented in this chapter assumed that only one change in the underlying distribution occurred. We were able to mitigate this in the 2D exploration simulation by caching any history of previously collected samples. However, it is possible that multiple changes could have occurred in the time series collected by the robot. Representing this, and dealing with an unknown number of changes would require more sophisticated machinery, but could be addressed by techniques like the multi-hypothesis developed by (Baum and Veeravalli, 1994), making for a logical extension of the work.

Similarly, we assume that the functions driving the background rate of the observations is constant in between changes. If that assumption does not hold up, then the hypotheses that are used to model the changes may not be able to do so correctly. For example, if the background rate was determined by a periodic function then a change detection algorithm assuming constant background rates would most likely frequently report that changes have occurred.

We assumed that the boundaries of changes in the underlying distribution are sharp, an assumption motivated by mission context in which this work was developed. There is a distinct boundary between different settings of the underlying distribution. A change that occurs more slowly over a long period of time may not be identified by the approach described in this chapter and would require a different competing hypotheses in the decision making algorithm. This is where an approach like the one described in Thompson et al. (2013) could be used as input to the decision making algorithm.

In the experiments we conducted in this chapter we assumed that the values reported from the sensor had infinite certainty. Accounting for uncertainty in the values would be a more principled (read: Bayesian) approach to change detection.

The changes in the distribution underlying the readings collected by the robot were considered to be a function of distance along the robot’s pre-determined path. This is considering the path to one dimensional. In actuality the robot is following a trajectory through (at least) two-dimensional space, and it should be possible to incorporate knowledge about the spatial relationships between readings to make better decisions. This does come at the cost of the simplicity of the algorithm discussed in this chapter and so trade-offs in computing resources should be considered in a mission context.

Because the scientists employing the neutron spectrometer have well characterized the relationship between the readings of the instrument and the abundance of subsurface water, there isn’t a need to learn that relationship on-line. However, one might be in the situation where the relationship between the prospecting sensor and the secondary sensor is not characterized and needs to be determined on-line. In this setting an upper confidence bound value function (Auer, 2003) like the one used by Das et al. (2015) would be more appropriate.

The decision rule used to determine if an AIM should be employed was a fairly trivial exemplar to illustrate the efficacy of the approach. Actual missions should employ AIM deployment decision rules that reflect mission risk and vehicle resources. Likewise the threshold for confidence in the change in the underlying distribution implies a certain risk tolerance. The confidence threshold should also be tuned to mission risk.

We do not consider the case of having a limited sampling budget. Should that become a constraint then the authors would recommend combining the change detection algorithm described
in this chapter with the submodular secretary described in Das et al. (2015).

The algorithm assumes that a path is pre-determined for the vehicle. In the 2D simulation we considered a lawnmower path of fixed width. The success and failure of the mission could depend on the spacing of switchbacks along that path. An informative path planner could improve the overall performance of the algorithm, reducing the risk of missing locations of interest in the world.

The SPRT algorithm as implemented searches through every time step in the data that have been collected from the last detected change. This is a linear search in the number of time steps, $T$, which in turn requires an order $T$ computation, yielding $O(T^2)$ complexity. However, we can reframe this problem as searching for the roots of the equation $\lambda(t) - s$, where $s$ is the confidence level threshold for choosing one hypothesis over the other. With this reframing we could employ, for example, a bisection search which would have a worst-case complexity of $O(T \log T)$.

### 4.5 Summary

The objective of the work in this chapter was to use a change detection algorithm to improve the deployment of discrete secondary sampling actions. We chose an Area of Interest Manoeuvre as the discrete, secondary sampling action. We designed an algorithm which is a straight-forward application of the sequential probability ratio test and compared it against a hard threshold technique which is used, in the author’s experience, by planetary scientists.

We have shown that the SPRT-based algorithm is able to out-perform the Memory Threshold, and Relative Change algorithms on real data from the MVP project, and in the simulated 2D environment. Most importantly, it is able to detect sub-threshold changes that would be invisible to an algorithm using a hard decision making threshold.

By recognizing trends that occur below the threshold, the autonomous science agent can make decisions about deployment that the competing algorithm can’t. By being able to detect changes with confidence the SPRT algorithm can reduce the rate of false positives, thus reducing waste of sampling resources.

The Relative Change algorithm is easily distracted by anomalies, resulting in a much higher false positive rate than the other algorithms. This behaviour could also be tuned to different scenarios, but again, it may need to be modified before every deployment. If the environment is truly unknown then this proposition is untenable.

In the simulated experiments we found that the Adaptive algorithm increases the number of changes detected, and reduces the error in estimating when the change occurs, compared to a fixed threshold algorithm. However, we found on the MVP data the improvements in performance were not maintained.

In the simulated 2D environment we found the SPRT algorithm was able to observe more maxima in the water maps than the Memory Threshold algorithm was, with potentially greater productivity. The number of maxima observed by the Adaptive Threshold was not statistically significantly different from the SPRT algorithm, but the Adaptive algorithm comes with the burden of additional parameters which must be tuned to the environment, the $\gamma_{\min}$ and number of suggested AIM deployments. We found with the MVP data that the $\gamma_{\min}$ parameter of the Adaptive Threshold algorithm still prevents sub-threshold activity from being detected.
It is difficult to recommend the use of the Adaptive algorithm, as the SPRT algorithm had better performance on efficiency, and because the Adaptive algorithm introduces more parameters than either the Memory Threshold or the Relative Change algorithm. There may certainly be benefits to encouraging the algorithm to deploy samples as a function of how much of the transect remains to be explored. To the author this seems to be conflating two notions, the first, determining how aggressively AIMs should be deployed, and second, what is the standard for an anomaly given environmental conditions. In our proposed algorithm we attempt to determine with some statistical confidence whether or not a change has occurred in the driving distribution, which seems to be a more direct solution to the problem.

Because the SPRT algorithm considers the likelihood that something has changed, and not the magnitude of that change, it is less dependent on prior knowledge for operation. With a reduced dependence on prior knowledge there should be greater confidence in performance when sending prospecting robots into unknown environments.
Chapter 5

Global Planning for Hypothesis Falsification

The previous chapters discussed methods for reactive autonomous science. Both the presented foraging and prospecting algorithms assumed that the trajectory to be followed already existed. In this chapter we develop an algorithm which determines locations in space to travel to in order to determine which of a set of competing hypotheses is most likely to be correct.

To say a hypothesis is correct relies on having a notion of truth. In a world with noisy sensors and variable processes having an objective notion of truth can be challenging, to say the least. In this work we use F.P. Ramsey's notion of truth, which is a gambler's notion of truth. That is to say, the thing which makes the predictions we can gamble on with greatest reward is the truth (Ramsey, 1931).

The hypotheses we consider in this chapter are functions that map observations in a domain, $X$, to a probability distribution over some other set of observations, $Z$. The task then set out for the learner is to collect sufficient data to determine the accuracy of a hypothesis, $h : X \rightarrow (Z \rightarrow [0, 1])$.

Previous approaches to science autonomy generally considers zero or one hypothesis at a time. Informative path planning attempts to collect observations in the domain of a hypothesis in question in order that an accurate model may be formed. The prototypical example of this is to find the distribution of some quantity over a region of a map. The algorithms that are used to design the sampling points come with certain assumptions about the smoothness and the continuity of the hypothesis over the domain.

In many respects this mirrors the notion of confirmation theory or verificationism, due to Carnap (1936). Verificationism was a bottom-up theory of science Hacking (1983), the experimenter was expected to collect data and generate hypotheses which fit those data. The belief was that sufficient observations enable one to generate hypotheses about the world from true statements regarding empirical data. Hypotheses would be proven true by the irrefutable weight of the experimental data.

Really this does not seem an objectionable stance. If one knew everything and had the time to sort through the data, then it should be possible to uncover the laws governing the operation of the world. This very much resembles function learning. One starts with a dataset and then attempts to identify the functions that best fit the data.
The space of functions searched is constrained by the representations used in the function learning process, and by the optimization and search techniques used to fit the function. These constraints are also added to help prevent over fitting, which is to say, to avoid functions that make accurate predictions where data has been collected, but that make wildly inaccurate predictions where data have not been collected.

Generalization is the holy grail of many a machine learning algorithm. The attempt to curtail over fitting, to ensure predictions on unseen data are reasonable, is a pre-emptive defense against David Hume’s problem of induction (Vickers, 2016). The problem of induction is that when extracting a law from observed data one really only knows that the law holds for those data, it says nothing about what may be encountered in the future.

Addressing the problem of induction is the motivating factor behind Karl Popper’s scheme of falsificationism (Popper, 2005). Falsificationism is based on the notion that no predictive statement about the world can be proven true, it can only be proven false. That is to say, that we may uncover that it’s predictive power is less useful in all circumstances than previously thought.

Under Falsification we can only consider hypotheses that make predictions about the state of the world. Further, those predictions must be, either directly or indirectly, observable. Then we can take the hypothesis, test its predictions and assess its relative truth.

With the best performing hypothesis we can then go about the world making predictions and reaping our rewards. Take for example the progression from Kepler’s Orbits to Galileo’s, or Newton’s physics to Einstein’s. Of course, it is possible that there might be a set of hypothesis which have different levels of predictive power at different points in the domain. The prototypical example here would be relativistic and quantum physics. They remain stubbornly ununited, so we keep both and apply them where they are most effective.

This is all well and good, but the question stands: What use is all this and how does it help us change how robot scientists behave? To answer that we have to look at what the implications of verificationism and falsificationism are for experiment design.

The way to design experiments under a verificationist regime is to try to observe as many points in the domain, \(X\), as possible. Greater information content is yielded by points that are farther away from each other, under whatever distance metric is used in the domain space. The logical end point of this thinking is to collect samples that are evenly spaced through the domain of the function. Under a falsification approach and in the case of testing only one hypothesis the end result is very similar. The objective is to take the predictions of the hypothesis and evaluate their accuracy. Then one must eventually test all parts of the hypothesis, leading to something resembling a uniform sampling across the domain of the hypothesis.

There may be more or fewer samples spent in locations correlated with the uncertainty in the predictions at that point. Sampling based on the uncertainty in the predictions of a function is a concept developed by Kristine Smith (Smith, 1918), the progenitor of mathematical experiment design, who preceded Popper. The underlying notion here is that predictions which are certain require fewer experiments to falsify than those which are uncertain.

Where falsification becomes more interesting is in the setting where there are multiple competing hypotheses and one wants to determine the best one. What becomes relevant in this setting is where the hypotheses disagree. Where hypotheses agree on their predictions they are all equally right or equally wrong, sampling in those regions does help assess their overall accuracy, but it does not help assess which of the hypothesis is better at making predictions. Points in
the hypotheses’ domains where they disagree in their predictions are exactly the points that one wants to sample to determine their relative accuracy.

Now obviously, if the goal is to assess the accuracy of different hypotheses, it is important to sample broadly in their domain in order and assess their overall performance. However, when there is the additional goal of simultaneously determining which of a fixed set of hypotheses is true, we should bias our search towards where they disagree. The combination of broad sampling in the domain of the hypotheses and favouring points of disagreement is the idea underpinning the algorithm developed in this chapter.

The proposed algorithm selects sample points in the domain of a set of competing hypotheses in order to determine which one of them is the most likely to be correct. We believe that our algorithm represents a principled fusion of verificationist and falsificationist style approaches to experiment design. Here we view falsification as sampling for information gain not in a distribution over the domain of a hypothesis, but information gain in the belief distribution over a set of competing hypotheses. It is the first planner to design navigation goals in order to evaluate competing hypotheses.

We test our algorithm in a simulated mission with a multi-hop lander that is trying to determine the accuracy of hypotheses that make predictions about some quantity that is not remotely observable. Such an activity might be drilling to access subsurface materials, or conducting laser ablation of the immediate environment. The point is, travel to and interaction with the environment is required. The domain of the hypotheses in this setting is something which makes predictions over a 2D map of the terrain.

This type of mission would be typical of a mission to, for example, the South Pole-Aitken Basin of the moon. Here the geological scale that must be sampled is on the order of 1000s of km, and as such would not readily admit a ground vehicle.

To further support the simulation findings we tested the algorithm in the Atacama desert as part of the ARADS project. The algorithm was deployed on the KRex2 robot where it chose locations to drill in order to localize subsurface halite.

The robot is testing a set of competing hypotheses that classify regions into one of two binary classes. Here we elide any intermediary features that might be used to classify the terrain and simply test binary maps as black-box hypotheses regarding some model presented to the robot from scientists.

Such hypotheses could be classifications developed from remote sensing data, such as the presence of frozen water in lunar soil, or subsurface halite in the Atacama desert. Depending on the models used to make predictions the same precursor data could generate multiple predictions about the underlying variable(s) of interest. The observations that robots collect on the surface help determine which of these hypotheses accurately predict what is observed on the surface, in turn implying the relative utility of the hypotheses.
Figure 5.1: A geological map of Jupiter’s Moon, Io. The remote data produced from observing Io has been interpreted as a map of terrain classifications. However, observations from the ground would be required to confirm the predictions of the map creators. Image courtesy wikipedia.org

We compare our algorithm to another algorithm which simply does mutual information sampling in the domain of the hypotheses. We find that our algorithm is statistically significantly better at identifying the best hypotheses out of the set than the competing algorithm.

The algorithm in question does not, however, use a planner that attempts to be efficient with any resource other than samples. The greedy planner simply hops to the next most informative site. Obviously this kind of approach would not scale to long-term operations with a ground vehicle, and we acknowledge that, along with the additional advantages to be gained by visiting locations between the identified goal points, however this algorithm could be used in such scenarios.

Additionally, the proposed algorithm does not attempt to produce its own hypothesis from the observations it collects. Nothing stops the algorithm from doing so. In fact, we maintain a parameter which tracks the relative belief that none of the presented hypotheses are correct. We find that by keeping track of this quantity for both the mutual information sampling and the proposed falsification sampling we are better able to have more confidence that none of the hypotheses are correct.

5.1 Prior Work

Science autonomy robots have not previously needed to choose between hypotheses, they have generally been focused on collecting data without prior knowledge. However, the design of experiments literature has been largely focused on selecting experiments that inform hypotheses.
A popular example of an experiment selection technique is the multi-armed bandit. Likewise, model selection is a well-studied field, but will not be reviewed here.

Multi-armed bandits (MABs) \cite{robbins1952some} are a formalization for sequentially selecting the most rewarding of a set of experiments. There are a number of different algorithms for approaching the MAB problem. The main families of which appear to Upper Confidence Bound algorithms \cite{lai1985asymptotically}, The Gittins Index \cite{gittins2011bandit}, and Thompson Sampling \cite{thompson1933likelihood}. Thompson sampling has recently gained attention for being simple to implement and for having near-optimal regret properties \cite{chapelle2011empirical, agrawal2012analysis, ortega2010}. Standard MAB settings assume the interaction with the candidate arms is simple. An arm is selected for testing, the arm is pulled, and a reward is delivered. In planetary exploration one can map candidate hypotheses to bandit arms, but executing tests is complex. Robots have to move through space and interact with the world, incurring unpredictable costs. Because targets of interest, like geologic materials, have highly spatially dependent distributions, sampling a random point is not guaranteed to be an informative action.

We look to science autonomy algorithms to determine how to select sample points. Most noteworthy of these algorithms is the OASIS/AEGIS \cite{castano2007oasis} system, as it has actually been deployed on Mars. However OASIS, like other prior work, either learn models from no prior data \cite{thompson2008wettergreen} or use proxy measures of importance that encode scientists’ preference\cite{paar2012, chien2005}, but are unaware of the hypotheses under examination. Given that access to data from precursor remote sensing is likely, starting each new scientific mission from a blank slate seems inefficient. 

Girdhar et al. \cite{girdhar2014}’s approach to exploration focusses on visiting locations that help construct a topic model for classifying satellite imagery. This approach is improves the topic model the robot is constructing but it does not inform any hypotheses that scientists are attempting to resolve.

Thompson and Wettergreen \cite{thompson2008wettergreen} maximize diversity of collected samples with mutual information sampling. This approach ensures samples are distributed through the input space of the function it is learning. A shortcoming of this approach, and all approaches that rely on mutual information sampling in Gaussian Processes with stationary kernel functions is that they do not depend on the observations collected. For a GP $P(f(x)|x_1, \ldots, x_t)$ the mutual information for a new data point, $x_{t+1}$, is proportional to:

$$ I(x_{t+1}) \propto k(x_{t+1}, x_{t+1}) - k_t(x_{t+1})^T K_t^{-1} k_t(x_{t+1}) $$ (5.1)

where $k$ is the kernel function. $k_t$ is the vector of distances between the $t$ collected data points and the test point $x$, and $K_t$ is the $t \times t$ matrix of the distances between all previously collected observations.

Mutual information sampling depends only on the points in the input space have been observed and the kernel function. The sequence of sampling points selected is deterministic once the initial point and the kernel function are known. While this is a principled approach, being aware of the hypothesis tested would require accounting for observations of $f(x)$.

The algorithm in \cite{thompson2015} is aware of observations made on the ground. Its robot attempts to explain spectra data from satellites by collecting a library of spectra from the
ground. Each pixel of satellite data is explained as a mixture of the end member spectra collected by the rover. The end members represent a basis of examples of “pure” minerals. The rover travels to locations where library of rover-observed spectra poorly explains the satellite data and collects more observations to explain the satellite data. This approach is implicitly constructing a hypothesis about the terrain composition but the algorithm itself does not test points in the satellite data that it considers well explained, nor does it consider alternate hypotheses.

Miller et al. (2016) uses the expected value of the Fisher Information to determine points of interest. Like mutual information (Lindley, 1956), Fisher information is used as a score to select the most informative experiments. Their path planner produces smooth paths maximize the number of high information value observations. Fisher information and the mutual information are intimately related, but not identical, quantities. However, a rigorous comparison of the behaviour of robots maximizing mutual information and those maximizing expected Fisher information does not exist in the literature. Without such evidence it is difficult to select one reward function over the other and this should be studied further.

Distributing points throughout the input space is a principled approach to learning a function de novo. However, when trying to select between competing hypotheses it is conceivable that a sequence of observation points $x_1, \ldots, x_t$ will miss the points of conflict between competing hypotheses, especially with limited sampling resources. However, widely sampling an input space is important to ensure the predictions of a hypothesis are generally accurate. Most deployed science autonomy algorithms do not operate in the realm of hypotheses, they either focus on distributions of data or on hand-coded proxy measures, not necessarily the support or falsification of hypotheses under consideration. For that reason we propose the method described in section 5.2.

The work that probably bears the greatest similarity to the work contained in this chapter is that of Balcan’s Agnostic Active Learning algorithm. Agnostic Active Learning, as developed in (Balcan et al., 2006), is an approach to selecting samples in order to determine which of a set of hypotheses most accurately models data. Through repeated sampling the algorithm reduces the number of hypotheses that are considered valid. The point of the algorithm is it repeatedly takes two hypotheses under consideration, determines where they disagree and sample in the region of disagreement. This is a method consistent with the strategy of falsification laid out by Popper. Where two hypotheses agree sampling there can not help determine which of the two hypotheses is more correct.

Their work, as described in (Balcan et al., 2006) focused on linear functions and threshold functions, which does make determining the region of disagreement more straight-forward than in the general case of functions. In our work we use a sampling strategy to determine which regions of the domain are interesting. While this has the downside of potentially considering more unnecessary locations, it has the distinct advantage of not requiring knowledge of how to compute the region of disagreement between competing hypotheses. Further, while the $A^3$ algorithm iteratively compares only two hypotheses at a time, we consider the change in belief over all the hypotheses. This comes at a greater computational cost, but it helps simultaneously re-evaluate more hypotheses.

The work contained in this chapter is an extension of the work presented in Furlong (2017). That work in turn bears a resemblance to the work presented in Dorsa Sadigh et al. (2017). The work by Dorsa et al. attempts to select the optimal control strategy for a dynamic system by first simulating performance and then asking a human, playing the role of an oracle, to rate
the performance of the algorithm. Both these algorithms are instantiations of the Thompson sampling algorithm.

The distinguishing factor between the work of [Dorsa Sadigh et al., 2017] and this work is where the former simply finds the best action at every iteration and then evaluates it. The work presented in this chapter seeks to take the sampling action, the question posed to the oracle in [Dorsa Sadigh et al., 2017], which yields the greatest reduction in uncertainty in the belief over the competing options.

Similarly, [Ravanbakhsh and Sankaranarayanan, 2017] engages in Thompson sampling-like behaviour. They are attempting to find control laws for different dynamic systems. However, where the work above attempts to learn accurately the relative performance of different points in the action space Ravanbakhsh et al. simply need to find one control law that is successful and halts immediately.

There have been proposed some algorithms for active model selection. These algorithms start with a fixed set of hypotheses and attempt to resolve which one of them are most accurate, given an oracle they can use to label points

[Madani et al., 2004] perform an empirical evaluation of active model selection algorithms. They consider a setting where the different hypotheses are modelled as coins, and the goal is to find the coin with the highest success rate. This differs from our problem in that the observations from one coin flip only informs the success rate of that coin. In our problem setting the observation from conducting a sampling action informs all the hypotheses under investigation.

However, [Madani et al., 2004] find that their “biased round-robin” sampling algorithm is one of the consistently high-performing algorithms. Biased round robin cycles through the coins in turn, but does not move from one coin to the next until a tails has been observed. This bears some similarity to our algorithm which treats as most likely the hypothesis which has been having the greatest success rate.

[Kumar and Raj, 2016] present a method for estimating the accuracy of a single, trained classifier with a limited sampling budget. They use the classifier to label the data, and then use those labels to segment the data into what strata. The goal was to estimate the accuracy of the classifier by requesting labels for the data in these strata.

They tested a number of different allocation techniques to allocate random samples to the different strata. They found that techniques which allocated samples proportional to the variance of the classifier accuracy within the strata performed well, but that an equal allocation of samples across the strata also had competitive performance.

[Kumar and Raj, 2016] use random sampling within the strata of data points. The algorithms they use are about deciding how many samples to draw from each location. It would be interesting to examine what happens when informative sampling is conducted within the strata.

[Ali et al., 2014] presents an active learning algorithm for simultaneously training and validating the different hypotheses, called Active Learning and Model Selection (ALMS). They consider candidate sampling points and score their relative value for training the hypotheses or selecting the best hypothesis.

In order to determine how valuable a candidate point would be for training, the algorithm predicts the label of the point by averaging over the predicted labels from all the hypotheses. ALMS then re-trains the hypotheses with the previously collected training set, all but two elements of the validation set, and the candidate point with its estimated label. Leave-two-out validation is
conducted with the remaining two data points, which have had labels determined by the oracle. The value of the candidate point is the average error in the cross-validation process, averaged by the current belief in the competing hypotheses.

The value of the candidate point for validation purposes is the average loss of the hypotheses trained with only the training dataset, computed using leave-two-out cross-validation on the entire validation set plus the candidate point with its estimated label. The training-value quantity is, in effect, asking which data point, if added to the training set, causes the worst performance for the validation of the hypotheses. On the other hand, the validation-value quantity is asking the question of which candidate point evinces the greatest reduction in belief in the hypotheses.

If the expected reduction in performance from training with the data point is higher than the reduction in validation, then the worst performing candidate point is added to the training data set. If the expected reduction in validation is greater than the reduction in training, then an unbiased sample is drawn from the pool of candidate points, and added to the validation data set.

The algorithm recomputes the probability of the models given the dataset it is building over every time step, and uses that to determine the value of previously unobserved sampling points. It is necessary to recompute this probability at before each sample is selected because the algorithm is simultaneously training the hypotheses with the data it collects while trying to validate them. In our approach, with fixed hypotheses, we don’t need to re-train the hypotheses, so we can accumulate evidence with each data point selected.

The algorithm proposed by Ali et al. (2014) does not account for the probability that none of the proposed hypotheses are correct, something we do address in this chapter. This is an important quantity to keep track of for future algorithms which will be not only searching for candidate sample points, but also new candidate hypotheses.

5.2 Method

The objective of the mission is to build belief in the hypothesis that is least false, $H^*$. In order to do this the robots must select locations, $l \in L$, in the input space of the hypotheses that most productively inform the investigation. The robots collect true observations, $M(l)$, from the true map, $M$, that can be used to update the belief in the different hypotheses based on the prediction $H_i(l)$ from the corresponding hypothesis $H_i$. The hypotheses the robots were attempting to validate were generated by corrupting generated map data. This is a stand-in for hypotheses that have different interpretations of the precursor data.

5.2.1 Belief in Hypotheses

The algorithms estimates their belief that hypothesis $H_i$, where $i \in 1, \ldots, K$, is the least false with a multinomial distribution, $P(H_i)$, where $\sum_{i=0}^{K} P(H_i) = 1$. We have a special place-holder hypothesis $H_0$, which represents the probability that none of the proposed hypotheses are correct.

We place a Dirichlet prior on the distribution of belief in the hypotheses, with corresponding condensation parameters $\alpha_0, \ldots, \alpha_K$. The condensation parameters are initialized with an uninformative prior, $\alpha_i = 1$, as initially all hypotheses are considered equally likely. However, the condensation parameters can be initialized to reflect any prior belief in the competing hypotheses.
After an observation the condensation parameters are updated to reflect the agreement with the predictions from the different hypotheses. That is to say:

\[ \alpha_{i,t+1} = \alpha_{i,t} + P(H(l(t)) = M(l(t)) | H_i) \] (5.2)

Initially we conduct experiments without uncertainty in the observations, and then complete experiments using noisy observations. However, we assume that the hypotheses themselves do not make probabilistic predictions. This mirrors the author’s experience working with planetary scientists.

5.2.2 Site Selection Algorithm

Algorithm 5.1 selects sampling sites with a modified version of Thompson sampling. At each time step, \( t \), the algorithm samples a belief state \( \hat{\theta} \) from the Dirichlet prior describe in Section 5.2.1 and then chooses the \( H_i \) that has the largest belief \( \theta_i \). The algorithm assumes that \( H_i \) is “true” for the duration of this step, and uses it to evaluate which location \( l_t \) is most informative. At the location \( l_t \) the robots collect the observation \( M(l_t) \). Observations are used to update the belief in the hypotheses that have been assigned to them.

We assume the robot starts in the \((0, 0)\) position on the map, which corresponds to the top left-hand location of the maps shown in Figure 5.2. The robot is capable of sampling at locations \( l \) in the input space \( L \), which in this experiment are \((x, y)\) locations on a two-dimensional map with integer coordinates. A Robot at location \( l_t \) that chooses to travel to \( l_{t+1} \) incurs a cost of \( \text{cost}(l_t, l_{t+1}) \). We assume that \( \text{cost}(l, l) = 0 \) and \( \text{cost}(l_1, l_2) > 0 \) when \( l_1 \neq l_2 \).

Once a location has been selected, the robot travels to the specified location and collects an observation from the actual map, \( M(l_t) \), and the belief state is updated. Should none of the hypotheses agree with the prediction of the map at \( l_t \), the place-holder condensation parameter \( \alpha_0 \) is incremented, increasing the belief that all hypotheses are equally false.

The reward function for candidate sampling locations is how the competing algorithms control the vehicle behaviour. The reward functions for the control algorithm and our proposed algorithm are specified in Sections 5.2.2 and 5.2.2.
Algorithm 5.1 Site Selection Algorithm

function SAMPLE-SELECTION($M, \langle H_1, \ldots, H_K \rangle, \text{budget}$)
\begin{align*}
\alpha_i & \leftarrow 1 \forall i \in 0, \ldots, K \\
t & \leftarrow 0 \\
l_0 & \leftarrow \langle (0, 0) \rangle \\
\text{repeat} \\
\bar{\theta} & \sim \text{Dirichlet}(\alpha_0, \alpha_1, \ldots, \alpha_K) \\
i & \leftarrow \arg\max_i \theta_i \\
l_t & \leftarrow \arg\max_i \text{reward}(\mathbb{E}[\bar{\theta}], H_i, \langle l_0, \ldots, l_{t-1} \rangle) \\
\text{for } i \leftarrow 1, K \text{ do} \\
\alpha_i & \leftarrow \alpha_i + 1 (M(l_t) = H_i(l_t)) \\
\text{end for} \\
\text{if } \sum_i^K 1 (M(l_t) = H_i(l_t)) = 0 \text{ then} \\
\alpha_0 & \leftarrow 1 \\
\text{end if} \\
\text{budget} & \leftarrow \text{budget} - \text{cost}(l_t) \\
t & \leftarrow t + 1 \\
\text{until } \text{budget} = 0 \\
\text{end function}
\end{align*}

Control Algorithm - Mutual Information Domain Sampling

For the control algorithm we build a density estimator using a Gaussian kernel function and we select the candidate point which the increase in information would be the greatest. We compute the entropy, $H(\cdot)$ of the kernel density estimator as described by Beirlant et al. (1997). The algorithm for the reward function is given in Algorithm 5.2.

Algorithm 5.2 Spatial Mutual Information sampling. This algorithm uses a measure of mutual information like the one used in (Thompson, 2008). It depends only on previously observed locations to find the most informative point.

function SPATIAL-REWARD($P(\bar{\theta}), H_i, l_0, \ldots, l_{t-1}, l_t$)
\begin{align*}
P_t(l) & \leftarrow \text{KDE}((l_0, \ldots, l_{t-1})) \\
\text{return } H(P_t) - H(P_t | l_t)
\end{align*}

Proposed Algorithm - Hypothesis Falsification Sampling

This algorithm seeks sampling locations that concentrate belief across the hypotheses given observation $H_i(l_t)$, as seen in Equation 5.3, where $H_i$ is considered the “true” hypothesis as per Algorithm 5.1. These locations maximize the mutual information between the belief state $\bar{\theta}$ and the observation $H_i(l_t)$.
\[ I(\vec{\theta}, H_i(l_t)) = \mathbb{H}[P(\vec{\theta})] - \mathbb{H}[P(\vec{\theta}|H_i(l_t))] \] (5.3)

This will automatically seek out locations that maximize the change in the belief distribution over the hypotheses. With Equation 5.3 as the reward function at any point \( l \) where the \( H_i \)'s agree will increase the credibility of all hypotheses, resulting in negative mutual information for \( l_t \).

However, one also wants to ensure that the hypotheses are accurate. Sampling at disagreement points will help build the credibility of the best hypothesis, but it does not give confidence with the overall predictions of \( H_i \). To mitigate this problem we add to Equation 5.3 the result computed in Algorithm 5.2 to ensure the sampled points spatially diverse. The reward function used for this algorithm is given in Algorithm 5.3.

We can derive this algorithm as follows:

\[
\mathbb{H}(\theta_i, L) = - \sum_i^N \sum_{l \in L} p(\theta_i, l) \log p(\theta_i, l)
\]

\[
= - \sum_i^N \sum_{l \in L} p(\theta_i|l)p(l) \left[ \log (p(\theta_i|l)p(l)) \right]
\]

\[
= - \sum_i^N \sum_{l \in L} p(\theta_i|l)p(l) \left[ \log (p(\theta_i|l)) + \log (p(l)) \right]
\]

\[
= - \sum_i^N \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] - \sum_i^N \sum_{l \in L} [p(\theta_i|l)p(l) \log p(l)]
\]

\[
= - \sum_i^N \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] - \sum_{l \in L} \left[ \sum_i^N p(\theta_i|l) \right] p(l) \log p(l)
\]

\[
= - \sum_i^N \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] - \sum_{l \in L} [p(l) \log p(l)]
\]

\[
= - \sum_i^N \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] + \mathbb{H}(L)
\]

\[
= - \sum_{l \in L} \sum_i^N [p(\theta_i|l)p(l) \log p(\theta_i|l)] + \mathbb{H}(L)
\]

\[
= \mathbb{E}_L \left[ - \sum_i^N p(\theta_i|l) \log p(\theta_i|l) \right] + \mathbb{H}(L)
\]

\[
= \mathbb{E}_L [\mathbb{H}(\theta_i|L)] + \mathbb{H}(L)
\]
Allowing us to conclude that
\[ H(\theta_i, L) = \mathbb{E}_L [\mathbb{H}(\theta_i|L)] + \mathbb{H}(L) \] (5.4)

The expectation term of Equation 5.4 can be difficult to compute as the space of \( L \) increases. However, since mutual information is the difference before and after observation the computation can be simplified.

\[ MI = H(\theta_i, L) - H(\theta_i, L|z(l_t), \xi = l_t) \] (5.5)

Substituting the derived value for entropy from Equation 5.4 into Equation 5.5 we can write
\[ MI = \mathbb{E}_Z [H(\theta_i|L = l_t)] - \mathbb{E}_Z [H(\theta_i|L = l_t, z(l_t), \xi = l_t)] + \mathbb{H}(L) - \mathbb{H}(L|z(l_t), \xi = l_t) \] (5.6)

Recognizing that \( H(L) - H(L|z(l_t), \xi = l_t) \) is the information gained by sampling at a location \( l_t \) given the prior belief state. The term \( \mathbb{E}_Z [H(\theta_i|L)] - \mathbb{E}_Z [H(\theta_i|L, z(l_t), \xi = l_t)] \) permits some opportunities to for simplification which all hinge on how much of the hypothesis an observation effects. Those options are:

**Global Effect** In this case we consider that every observation contributes to the average belief in the hypotheses. In this case we do not propagate belief over the input space but simply accumulate evidence to give credibility to the hypotheses.

**Naive Assumption** In this case we believe that the belief in the hypotheses at the location \( l_t \) is independent at all other locations. Here we simply need to compute how the belief at this location changes. This is an unsatisfying choice as it cannot accumulate the evidence from all other previous beliefs and simply gives credit for observing locations that have not been observed before.

**Neighbourhood Effects** If there is a relationship over the input space \( L \), some manner of smoothness in the belief state over \( L \) then if one models it with a limited range effect (e.g. a Gaussian Process using a covariance kernel with finite support)

**Non-local Effects** An observation at any location \( l_t \in L \) effects the belief state at all other locations \( l' \in L \).

**Algorithm 5.3** Hypothesis Falsification sampling. We use Equation 5.3 to increase the value of spatially diverse points

```plaintext
function DISAGREE-REWARD(P(\bar{\theta}), H_i, \langle l_0, \ldots, l_{t-1}, l_t \rangle)
    P(\bar{\theta}|H_i(l_t)) \leftarrow \text{update}(P(\bar{\theta}), H_i(l_t))
    r_h \leftarrow \mathbb{H}[P(\bar{\theta})] - \mathbb{H}[P(\bar{\theta}|H_i(l_t))]
    r_s \leftarrow \text{spatial-reward}(P(\bar{\theta}), H_i, \langle l_0, \ldots, l_{t-1}, l_t \rangle)
    return r_h + r_s
end function
```
5.2.3 Map Generation

We generated 10 different maps, each $20 \times 20$ pixels, with each pixel containing a label from $z \in 1, \ldots, N$. We produce the ground truth maps by selecting 20 seed locations and randomly assigning them a label $z$ with uniform probability (denoted by the function $U(\cdot)$), over the $N$ labels. We then use a Voronoi map generation algorithm, given in Algorithm 5.4. Example maps are shown in Figure 5.2a.

Algorithm 5.4 Map Generation Algorithm

$$
\text{function GENERATE-MAP(numSeeds, numPixels, numLabels)}$

$$
\text{seeds} \leftarrow \langle \cdot \rangle$

$$
\text{map} \leftarrow \text{zeros}(\text{numPixels, numPixels})$

$$
\text{for } i \in 1, \ldots, \text{numSeeds} \text{ do}$

$$
\text{x}_i \sim U(\text{numPixels})$

$$
\text{y}_i \sim U(\text{numPixels})$

$$
\text{label} \sim U(\text{numLabels})$

$$
\text{seeds} \leftarrow \text{seeds} + \langle (\text{x}_i, \text{y}_i, \text{label}) \rangle.$$

$$
\text{end for}$

$$
\text{for } x \in 1, \ldots, \text{numPixels} \text{ do}$

$$
\text{for } y \in 1, \ldots, \text{numPixels} \text{ do}$

$$
\text{map}(x, y) \leftarrow \text{closest}(\text{seeds}, x, y).\text{label}$

$$
\text{end for}$

$$
\text{end for}$

$$
\text{return map}$

$$
\text{end function}$

To generate the hypotheses maps we take the same 20 seed points used to generate a true map and mislabel the seed locations with probability $P(z = i | z = j) = \epsilon/(N - 1)$ for all $i \neq j, i, j \in 1, \ldots, N$. Using the corrupted seeds we then generate maps using the Algorithm 5.4. Figures 5.2b, 5.2c, 5.2d shows the effect of maps as $\epsilon$ increases. We tested hypotheses generated with $\epsilon \in \{0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9\}$ which correspond to $H_1$ to $H_7$ in Table 5.1. For simplicity of experiments we set $N = 2$ and assumed no sensor noise on the part of the robot, but again the algorithm admits more complex scenarios.

5.2.4 Experiments

We compared the performance of our proposed algorithm and of the control algorithm in three experiments. For each experiment we tested ten different maps, and each map was repeated twenty-four times. The true maps and the hypothesis maps were generated as described in Section 5.2.3. The similarity between the true map and the hypotheses are given in Table 5.1. When a robot travelled to a point on the map it was informed of the true classification of the point with no error.
Figure 5.2: Examples of a map and the hypotheses generated that anticipate the map.

<table>
<thead>
<tr>
<th>True Maps</th>
<th>$H_0$</th>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$H_3$</th>
<th>$H_4$</th>
<th>$H_5$</th>
<th>$H_6$</th>
<th>$H_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>0.93</td>
<td>0.82</td>
<td>0.66</td>
<td>0.49</td>
<td>0.26</td>
<td>0.14</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td>0.91</td>
<td>0.86</td>
<td>0.70</td>
<td>0.53</td>
<td>0.24</td>
<td>0.11</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>0.90</td>
<td>0.79</td>
<td>0.73</td>
<td>0.62</td>
<td>0.23</td>
<td>0.19</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>0.95</td>
<td>0.86</td>
<td>0.77</td>
<td>0.36</td>
<td>0.25</td>
<td>0.23</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>0.98</td>
<td>0.81</td>
<td>0.74</td>
<td>0.67</td>
<td>0.23</td>
<td>0.20</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td>1.00</td>
<td>0.83</td>
<td>0.69</td>
<td>0.45</td>
<td>0.29</td>
<td>0.18</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>$M_6$</td>
<td>0.98</td>
<td>0.91</td>
<td>0.86</td>
<td>0.42</td>
<td>0.20</td>
<td>0.17</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>$M_7$</td>
<td>1.00</td>
<td>0.80</td>
<td>0.79</td>
<td>0.55</td>
<td>0.38</td>
<td>0.16</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>$M_8$</td>
<td>0.94</td>
<td>0.77</td>
<td>0.76</td>
<td>0.62</td>
<td>0.30</td>
<td>0.10</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>$M_9$</td>
<td>0.99</td>
<td>0.92</td>
<td>0.76</td>
<td>0.42</td>
<td>0.20</td>
<td>0.14</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: This table gives the similarity between the true maps $M_i$ and the hypotheses proposed by the simulated scientists $H_j$.

We assume in this experiment that the robot is capable of hopping long distances. As such the cost of traversal is simply limited to the number of hops. The robots were given a budget of $T \in \{25, 50, 100, 150, 200\}$ hops. Since the mission budget is in number of hops the cost function is the constant function $\text{cost}(\cdot, \cdot) = 1$. The budget and cost function can be modified to different mission settings.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>$H^*$</th>
<th>Hypotheses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - All Good Maps</td>
<td>$H_1$</td>
<td>$H_1$ $H_2$ $H_3$</td>
</tr>
<tr>
<td>2 - Mixed Quality Maps</td>
<td>$H_1$</td>
<td>$H_1$ $H_4$ $H_7$</td>
</tr>
<tr>
<td>3 - All Bad Maps</td>
<td>$H_0$</td>
<td>$H_5$ $H_6$ $H_7$</td>
</tr>
</tbody>
</table>

Table 5.2: The hypotheses tested in the experiments. Column $H^*$ gives the best hypothesis for each experiment. Subscripts correspond to Table 5.1.

In Experiment 1 the hypotheses are all of reasonable quality. This represents a mission with good precursor data and generated hypotheses. In Experiment 2 the hypotheses are of mixed
quality. In this scenario the precursor data is ambiguous, but there is one good hypothesis. In Experiment 3 all hypotheses are of poor quality. In this scenario the hypotheses do not describe the environment accurately.

**Analysis of Results**  At the end of the experiments we compute mean of $P(H^*)$. Across the ten different maps we computed the effect size of our intervention. To determine the effect size we use Cohen’s $d$, with pooled standard deviation, $\sigma_{pooled}$.

$$
d = \frac{\mu_1 - \mu_0}{\sigma_{pooled}}
$$

(5.7)

$$
\sigma_{pooled} = \sqrt{\frac{\sigma_1^2 + \sigma_0^2}{2}}
$$

(5.8)

where $\mu$ and $\sigma$ are the sample mean and standard deviation of $P(H^*)$ for the control (0) and proposed (1) algorithms, respectively. Cohen’s $d$ has four levels for effect size: $d < 0.2$ is a negligible effect, $0.2 < d < 0.5$ is a small effect, $0.5 < d < 0.8$ is a medium effect, and $d > 0.8$ is a large effect. Negative values of $d$ mean that the control algorithm has a higher belief in $H^*$. There are criticisms of using these standards Lipsey et al. (2012), but the dearth of effect size reporting in robotics does not provide alternative thresholds for effect size magnitudes.

In all bar charts a double star marker, (**), indicates that the probability that the difference between the distributions is non-zero is greater than or equal to 95%.

### 5.3 Results

We break the results up based on whether or not noisy observations were included in the experiment.

In the tables below we denote whether statistical significance has been achieved with $p \geq 95\%$. We use paired Bayesian significance testing, as per Bååth (2014). To report effect size we use Cohen’s $d$ (Lipsey et al., 2012).

#### 5.3.1 Hypothesis Falsification with Noise-Free Observations

Figure 5.3 illustrates the average belief over the hypotheses for the three different conditions with a budget of 50 samples. We can see that both mutual information and falsification distribute their belief across the hypotheses in a way that reflects the general accuracy of the hypotheses. However, as we will see below, the falsification sampling has an advantage in belief in the best hypotheses.
Figure 5.3: Plots of the belief in the hypotheses in the three different experiments without sensor noise. These results are for sampling budget of 50, and the behaviour is averaged over the 10 true maps.

For detailed plots of average performance of the algorithms for all budget settings see appendix D.

Experiment 1 Results - All Good Hypotheses

In this experiment the correct hypothesis to select is \( H_1 \).

Table 5.3

<table>
<thead>
<tr>
<th>Budget</th>
<th>( \Delta P(H^*) )</th>
<th>( p \geq 0.95 )</th>
<th>Cohen’s ( d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>2.3%</td>
<td>Y</td>
<td>0.47</td>
</tr>
<tr>
<td>50</td>
<td>2.5%</td>
<td>Y</td>
<td>0.72</td>
</tr>
<tr>
<td>100</td>
<td>2.2%</td>
<td>Y</td>
<td>0.69</td>
</tr>
<tr>
<td>150</td>
<td>2.3%</td>
<td>Y</td>
<td>0.72</td>
</tr>
<tr>
<td>200</td>
<td>2.5%</td>
<td>Y</td>
<td>0.78</td>
</tr>
</tbody>
</table>

We can see that the increase in belief in the best hypothesis occurs for all budget sizes. The average change in belief is approximately 2.4\%, with a medium effect size. While the change is small, the effect size shows that the results are real.

Experiment 2 Results - Mixed Quality Hypotheses

In this experiment the correct hypothesis to select is \( H_1 \).
Table 5.4

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\Delta P(H^*)$</th>
<th>$p \geq 0.95$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>2.9%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>4.8%</td>
<td>Y</td>
<td>0.63</td>
</tr>
<tr>
<td>100</td>
<td>4.5%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>150</td>
<td>4.9%</td>
<td>Y</td>
<td>0.67</td>
</tr>
<tr>
<td>200</td>
<td>5.4%</td>
<td>Y</td>
<td>0.75</td>
</tr>
</tbody>
</table>

When the quality of the hypotheses are mixed, one good, one moderate, and one poor quality hypothesis, both algorithms pick the best available hypothesis. Falsification search has greater belief in the best hypothesis, with an average increase of belief about in the range of 4.4% to 5.4%, with probability $> 95\%$, and medium effect size.

Experiment 3 Results - All Bad Hypotheses

In this experiment the correct hypothesis to select is $H_0$.

Table 5.5

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\Delta P(H^*)$</th>
<th>$p \geq 0.95$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>17.8%</td>
<td>Y</td>
<td>0.92</td>
</tr>
<tr>
<td>50</td>
<td>16.6%</td>
<td>Y</td>
<td>0.94</td>
</tr>
<tr>
<td>100</td>
<td>13.6%</td>
<td>Y</td>
<td>0.78</td>
</tr>
<tr>
<td>150</td>
<td>11.0%</td>
<td>Y</td>
<td>0.81</td>
</tr>
<tr>
<td>200</td>
<td>10.3%</td>
<td>Y</td>
<td>0.75</td>
</tr>
</tbody>
</table>

In the case where all the hypotheses are bad the falsification sampling algorithm is better at identifying that none of the hypotheses are correct is greater than the mutual information sampling, which is the desired behaviour in this case. Further, the improvement is statistically significant and the effect size is large.

We note that as the number of samples are collected the performance tends towards the behaviour of the mutual information sampling algorithm. However, for the range of budgets sampled, falsification sampling maintains it’s superiority.

5.3.2 Hypothesis Falsification with Noisy Observations

When sensor noise was introduced the different in belief distribution learned by the mutual information sampling and the falsification sampling was reduced, as can be seen in Figure 5.4. Again, this figure is for the algorithms when the sampling budget was 50 samples. As in the noiseless scenario, both mutual information and falsification distribute their belief across the hypotheses in a way that reflects the general accuracy of the hypotheses. We can see that the behaviour of the falsification algorithm gracefully degrades to the behaviour of the mutual information sampling.
Figure 5.4: Plots of the belief in the hypotheses in the three different experiments with sensor noise. These results are for sampling budget of 50, and the behaviour is averaged over the 10 true maps.

For detailed plots of average performance of the algorithms for all budget settings see appendix D.

Experiment 1 Results - All Good Hypothesis

In this experiment the correct hypothesis to select is $H_1$.

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\Delta P(H^*)$</th>
<th>$p \geq 0.95$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.3%</td>
<td>Y</td>
<td>0.80</td>
</tr>
<tr>
<td>50</td>
<td>0.2%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>100</td>
<td>0.2%</td>
<td>Y</td>
<td>0.65</td>
</tr>
<tr>
<td>150</td>
<td>0.2%</td>
<td>Y</td>
<td>0.63</td>
</tr>
<tr>
<td>200</td>
<td>0.1%</td>
<td>Y</td>
<td>0.68</td>
</tr>
</tbody>
</table>

With the introduction of noise to the sensors we see the gap in performance has decreased between the falsification sampling and the mutual information sampling. However, in all cases the effect size was at least medium. Both algorithms made the correct choice of hypothesis, but the falsification sampling had a slight, but real, advantage.

Experiment 2 Results - Mixed Quality Hypothesis

In this experiment the correct hypothesis to select is $H_1$. 

Belief vs Hypotheses: $H_0$, $H_1$, $H_2$, $H_3$
Table 5.7: $H^* = H_1$

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\Delta P(H^*)$</th>
<th>$p \geq 0.95$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>-0.2%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>0.1%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>100</td>
<td>0.1%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>150</td>
<td>0.1%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>200</td>
<td>0.2%</td>
<td>Y</td>
<td>0.63</td>
</tr>
</tbody>
</table>

In all but the largest sampling budgets the performance difference between falsification sampling and mutual information sampling are statistically indistinguishable at a level of 95% confidence. When the budget is 200 samples the falsification algorithm regains a slight advantage on the mutual information sampling. Both algorithms select the best hypothesis for all budget sizes.

**Experiment 3 Results - All Bad Hypotheses**

In this experiment the correct hypothesis to select is $H_0$ - the hypothesis that none of candidate hypotheses are correct.

Table 5.8: $H^* = H_0$

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\Delta P(H^*)$</th>
<th>$p \geq 0.95$</th>
<th>Cohen’s $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1.4%</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>1.7%</td>
<td>Y</td>
<td>1.14</td>
</tr>
<tr>
<td>100</td>
<td>1.0%</td>
<td>Y</td>
<td>1.03</td>
</tr>
<tr>
<td>150</td>
<td>0.6%</td>
<td>Y</td>
<td>1.08</td>
</tr>
<tr>
<td>200</td>
<td>0.7%</td>
<td>Y</td>
<td>1.91</td>
</tr>
</tbody>
</table>

With the introduction of noise we see a closing of the gap in the belief that no hypothesis is correct, by approximately an order of magnitude. This order of magnitude is about the same probability of the error in the noise, so we would anticipate this magnitude of change. What is interesting is that the effect size here has increased, meaning that while the difference in belief has changed, the variability of that change has also reduced.

In the experiments with noisy observations we see that the falsification is able to maintain a statistically significant difference in some cases, and in all cases it selects the correct hypothesis. We can take comfort in the fact that when noise is present the performance of the falsification sampling approach gracefully decays towards that of the mutual information sampling. The implication being here, that all things being equal, selecting falsification sampling is never a worse than using standard mutual information sampling, and can be significantly better.

### 5.4 Deployment in Chile

To demonstrate the proposed algorithm we deployed it on the KReX2 robot in Chile (Figure 5.5) as part of the 2017 field season of NASA Ames’ ARADS project. Planetary scientists who were
part of the project were interested in mapping subsurface halite deposits. The scientists considered a location around a region they called “the pit”, which was located, in UTM coordinates, at Zone 19J 396528.98 E 7334167.93 S.

![Image of KRex2 robot with mounted drill and sample collection arm exploring the Atacama Desert in Chile. The drilling process could take up to four hours to drill a 2m deep hole, including time to operate sample collection arm.](image)

**Figure 5.5:** The IRG robot KRex2 with a mounted drill and sample collection arm exploring the Atacama Desert in Chile. The drilling process could take up to four hours to drill a 2m deep hole, including time to operate sample collection arm.

The scientists were interested in two competing hypotheses about the distribution of subsurface halite. They had two competing hypotheses that they were interested in. The first hypothesis ($H_1$) was that there is a uniform distribution of halite in and around the pit. The competing hypothesis was that the halite was a deposit resulting from a dried body of water in approximately 30m in diameter, centred at the pit ($H_2$). We maintained both these hypotheses along with the belief that neither hypothesis was correct. The domain of operation was restricted around the pit to field of operations enclosing that 30m space. The two hypotheses are illustrated in Figure 5.6.

The subsurface halite was detected using a rotary-percussive drill developed by Honeybee, Inc. [Bergman et al., 2016](#). The halite was detected by having a human operator observe the current draw of the primary motor in the drill. When current draw had crossed a boundary the human observer determined that halite had been encountered. Obviously this is a less than ideal sensor and not rigorously calibrated. However, laboratory analysis later determined that halite was recovered from the holes where the current draw indicated that halite was discovered.

It took approximately one hour to reach a depth of 1m. Since the halite was generally known to be at a depth of 1.5m in the area hills were drilled to a depth of 2m. This took approximately
two hours to drill to this depth, with additional time to extract the drill from the hole after a maximum depth was reached. Five drilling sites were identified by the algorithm and were then drilled before the allotted time for the exercise was exhausted. Those locations and the finding from the drill are given in Table 5.9.

We discretized the operations area into a 1m resolution grid. At each time step the algorithm considers which point in the grid would be most informative about the two hypotheses. When a grid point is selected the robot travels there and commences drilling. The presence of halite was determined by the human observer, and this observation was fed into the algorithm. The holes were drilled in the order number given in Table 5.9.

Table 5.9: The resultant observations from drilling operations in the Atacama. Holes were selected in the order given by the column titled “Hole”.

<table>
<thead>
<tr>
<th>Hole</th>
<th>Commanded Hole Coordinates (UTM)</th>
<th>In Circle?</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zone 19J 396515 E 7334151 N</td>
<td>No</td>
<td>Halite present</td>
</tr>
<tr>
<td>2</td>
<td>Zone 19J 396548 E 7334182 N</td>
<td>No</td>
<td>Halite not present</td>
</tr>
<tr>
<td>3</td>
<td>Zone 19J 396514 E 7334182 N</td>
<td>No</td>
<td>Halite not present</td>
</tr>
<tr>
<td>4</td>
<td>Zone 19J 396543 E 7334153 N</td>
<td>No</td>
<td>Halite not present</td>
</tr>
<tr>
<td>5</td>
<td>Zone 19J 396531 E 7334166 N</td>
<td>Yes</td>
<td>Halite present</td>
</tr>
</tbody>
</table>

In Figure 5.7 we can see the belief in the different hypotheses change as a function of the
holes drilled. As the number of holes are increased the belief in the second hypotheses is increased, supported by the evidence collected from the drill samples. The locations selected by the algorithm focused on where the competing hypotheses disagreed and focused its initial samples there. The final sampling point was located at the centre point of the domain. Presumably the component of the reward function that attempts to maximize the diversity of the algorithm forced the sample point to be the location furthest from the other samples while still staying within the domain of operation.

Evolution in Belief in Competing Hypotheses as Data are Acquired from Drill Holes

![Graph showing belief evolution in competing hypotheses](image)

**Figure 5.7:** This plot shows the belief in the different hypotheses changing over time. Ultimately the non-uniform hypothesis was found to be most credible. This algorithm sent the robot first to points of disagreement between the two hypotheses.

On the whole the behaviour of the algorithm in the field was consistent with the performance
demonstrated in the preceding experiments. The amount of time it took to drill a hole and the restricted operation time for this portion of the field season drastically limited the number of holes that could be drilled. However, even though the number of samples was drastically limited compared to the simulation experiments the algorithm still behaved accordingly.

5.5 Discussion

In this chapter we considered only one noise model. While we would expect to observe similar results, degradation of performance towards that of the mutual information sampling, it would be worth supporting this supposition with more and varied experimentation.

Noisy observations would also warrant taking multiple readings at different sites to reduce the effect of that noise. A principled strategy for trading off the reduction in uncertainty and the mission costs would be an important addition to a deployed system. This could be done, albeit naively, by allocating a fixed number of samples to each sample site in order to reach an acceptable confidence level.

We do not consider hypotheses that make probabilistic predictions. This representation is consistent with the author’s experience with planetary science, but it is unsatisfying from a Bayesian standpoint. Incorporating such hypotheses is not challenging given the framework presented in this chapter and it should be considered in future work.

Similarly, the range of the predictions made by the hypotheses was quite narrow, it being the set \( \{0, 1\} \). Hypotheses which make a broader range of predictions may yield more profound differentiation in relative belief.

The exhaustive evaluation of the domain of the hypotheses when attempting to identify the best location to sample does not scale well with the size of the domain. Scaling would also be a problem if the extent of the domain remains the same, but the level of discretization is increased. A sampling strategy, such as a probabilistic roadmap, would mitigate the increased computational demand at the expense of some accuracy.

There are at least three obvious next steps that should be taken to continue this aspect of the thesis. First, do falsification planning that is more aligned to ground vehicle navigation. Second, take techniques from computational learning theory to control when hypotheses are rejected and new ones are sought. Thirdly, integrate hypothesis generation algorithms.

The greedy algorithm used to select the waypoints in not necessarily ideal. It assumes that the vehicle has random access to any point in the hypotheses’ domain. This assumption holds in the narrow context of the hypothesized mission discussed in the chapter, and indeed for experiment design where moving between experiments doesn’t have a traverse cost, such as in drug design. But in the case of ground vehicles this assumption is not valid.

Paths for ground vehicles that ignore all the intervening locations between two goals is missing out on potentially valuable sampling opportunities. However, estimating the value of different trajectories can become computationally expensive. For that reason it would be natural to use an approximate planner like the one used by Arora et al. (2018), and faster information gain measures like the Cauchy-Schwartz quadratic mutual information used by Tabib et al. (2016).

The second improvement would be to use techniques from computational learning theory to evaluate the performance of the different hypotheses. Specifically, using concepts like Proba-
bly Approximately Correct bounds to estimate whether or not a hypotheses can be found to be accurate within a certain error threshold given the number of samples that are available.

What would make the PAC bounds even more useful is to develop a means to estimate the number of samples required to estimate the effectiveness of a hypothesis given its symbolic representation and the anticipated noise models of the sensors involved in sampling. Much like automatic differentiation has been a game changer for artificial neural networks, such an automatic learning bound estimate would be instrumental in automated experiment design.

There is a problem with using PAC bounds, however. They rely on arbitrarily set thresholds for how accurate a hypothesis should be, and with what level of confidence we are prepared to accept that accuracy. These must come from somewhere and could change how aggressively an algorithm would search the space of possible hypotheses. At this point the author cannot think of an objective means to control these parameters.

Finally, the coupling of this algorithm with hypothesis generation algorithms is a vital part of liberating the robot from human oversight. The main objective of including the $h_0$ hypothesis in the algorithm was to identify when new hypotheses are needed. Using this belief state we could trigger the search for better hypotheses given the data collected so far, and add them to the set to be evaluated.

Granted, having a robot simultaneously generate and falsify hypotheses while exploring the world is going to require substantially larger sampling budgets than are typically deployed in space missions. It is the necessary step to close the cycle of science in a deployed, fielded robot, a larger objective than just flight missions.

The approach developed in this chapter reduces the need to have a reliable communications link between the robot and remote human scientists. Since the algorithm is able to identify when the presented hypotheses are not credible, combining this algorithm with some of the hypotheses generation algorithms discussed in chapter 2 would be a good step towards freeing robots from a dependence on humans altogether. A robot with the capacity to generate and evaluate hypotheses, and determine when to ask for new hypotheses, could conceivably conduct long-term scientific exploration without direct human supervision. Simultaneous hypothesis generation and falsification represents an exciting new direction for science autonomy research.

5.6 Summary

In this chapter we set out to develop an algorithm that sampled the domain of a set of hypotheses in order to determine which hypothesis was the most accurate. We achieved this by doing mutual information sampling in the belief space over the hypotheses, in addition to mutual information sampling in the domain of the hypotheses.

We found that our approach produced a statistically significant improvement in performance, which tends to increase as the number of samples increases. While this relationship held both with and without noisy sensors, the gap in performance degraded with sensor noise.

The proposed algorithm never underperforms the baseline, and gracefully reverts towards the results of mutual information sampling in the hypotheses’ domain. Likewise, should it become possible to exhaustively sample the domain of the hypotheses then the proposed falsification sampling should converge with the domain-only mutual information sampling. However, if it
becomes possible to exhaustively sample the domain of the hypotheses, then the problem that our proposed algorithm solves vanishes.

Because we use a sampling approach to evaluating different parts of the hypotheses’ domain, we do not need to understand the structure of the hypotheses in order to select informative sampling points. This gives us an advantage over the approach developed by (Balcan et al., 2006). Since the functional form of the hypotheses is not necessary to evaluate their accuracy the proposed algorithm can, and does, handle hypotheses which are more complex than simple linear classifiers.

In the presented experiments when all of the hypothesis are good, or when there is one clear winner, our algorithm performs at least as well as mutual information sampling in the hypotheses’ domain. When all the hypotheses were bad, our algorithm is better at identifying that that is indeed the case.

When noise is introduced, performance decays towards the performance of the mutual information sampling. This gives us confidence that it is safe to deploy this algorithm in situations where mutual information sampling would already be deployed.

Given that a long-term objective for science autonomy is to have robots conducting science without human supervision, the increased ability to identify that no hypotheses are correct is vital. If the robot needs to be able to identify that a new hypothesis is needed in order to decide when to ask for, or generate, a new one.

Previous approaches to science autonomy simply sample in the domain space. Others use proxy functions for the values that the scientists are interested in. There is the danger that coding for targets of interest may result in hard-coding confirmation bias into the robot.

This work introduces, to the best of the author’s knowledge, the first robot action planner which attempts to test multiple, competing hypotheses. Using techniques like the one proposed, robots can use humans’ hypotheses to direct their work, without further guidance from humans.
Chapter 6

Conclusion

In this thesis we set out to improve three aspects of autonomous exploration by accounting for operational conditions. Each of these algorithms – foraging, prospecting, and falsification sampling – represent components of a robot conducting science autonomously. We have shown, with statistical significance and at least a moderate effect size, in all cases that improvements in performance were achieved.

We were able to improve the performance of foraging for information by recognizing that exploring agents do not have random access to all objects they would like to sample. We further demonstrated that distributions learned by foraging can be improved by detecting changes in the distributions underlying classes of objects being sampled. We improve the performance of the prospecting algorithm by building confidence in detected changes in scalar field observed by the proxy sensor. Finally, we determined that falsification based sampling can more effectively identify which of a set of hypotheses most credibly explains the data. Below we lay out the specific contributions of this work, discuss limitations of the work, and then review possible avenues of future work.

6.1 Contributions

Our approach touched on three different aspects of autonomous field science. We demonstrated a causal link between the proposed algorithms and improvements over the credible baseline algorithms all three approaches. The first two algorithms deal with situations where the robot is operating without global knowledge, the third algorithm plans experiments to determine which of a set of hypotheses are the most credible.

**Foraging** We considered opportunistic science when sampling discrete objects which are randomly distributed about an environment. Previous approaches did not consider the distribution governing which classes of objects an exploring robot would encounter. We have shown that for a variety of sampling and exploration costs our algorithm produces a statistically significant reduction in error in the estimation of the underlying distribution.

We also found that, given sufficient sampling budget, there were different settings of sampling and exploration costs which resulted in the foraging, greedy, and uniform sampling having the
best performance. When the exploration and search costs are low, the foraging algorithm is the best algorithm with statistical significance. To the best of the author’s knowledge this is the first application of foraging algorithms to opportunistic sampling.

Further, we have identified, that there are scenarios where the max greedy and uniform sampling algorithms can have a statistically significant improvement over the other tested algorithms. By knowing the performance of the algorithms given exploring and sampling costs, and an operational budget, a robot scientist could modify its behaviour to produce better results.

Conversely, if one knows exploration costs, then one could design sampling operations to control the costs of sampling, and keep robot performance within a desired regime. However, a formal understanding of these relationship remains to be determined.

We have also demonstrated that, with enough observations, a robot sampling discrete objects can successfully detect changes in the underlying distribution. This algorithm permits the robot to monitor changes while learning about the different classes of objects. This permits the robot to reset its probability distribution estimates and to segment a map based on their observations, which may help demarcate changes in environment. Regardless of the opportunistic sampling algorithm that is used by a robot, change detection would make a useful addition to any opportunistic sampling algorithm.

Prospecting: The shortcoming that we identified in prospecting algorithms as they are practiced was the reliance on hard thresholds. In our work we demonstrated that we can estimate the confidence that the distribution underlying the readings have changed. The state of the art in this area is to deploy more expensive actions when readings have crossed a threshold. Our approach recognizes what could be sub-threshold behaviour, and need not be tuned to every environment.

When applied to the task of localizing maxima in a scalar field, and compared to the threshold-based algorithm we were able to better localize the maxima in the field with statistical significance and large effect size. While we did not capture as many maxima in the field as a surprise-based algorithm, change-detection algorithm made more effective use of the AIMs we deployed, with statistical significance and large effect size.

Our algorithm did not need to be adjusted for different deployments, unlike threshold based algorithms. We demonstrate on real data that our algorithm was able to detect sub-threshold changes in the distribution and act accordingly. This work represents a new, simple, algorithm that can be deployed in space hardware, and is agnostic of the trajectory the robot is following.

Falsification: The third contribution of this thesis is an action planner that aims to determine which of a fixed set of hypotheses is most accurate. The algorithm selects actions that inform not only the belief distribution over the hypotheses, but also the performance of the hypotheses themselves. We compare the algorithm against mutual information sampling and find a statistically significant improvement.

Our algorithm does not need to understand the functional form of the hypotheses being investigated. It does require that it is possible to evaluate the hypotheses at arbitrary points in their domain. Because we used a sampling approach to evaluating the value of exploratory actions the robot only needs to know the predictions of the hypotheses and not their structure or functional form.
We found that the proposed algorithm was better able to identify that none of the hypotheses were correct when all the hypotheses had less than 50% accuracy. This implies that the algorithm is better able to determine that a new hypothesis is needed when none of the proposed hypotheses are suitable.

We demonstrated this algorithm on the KReX2 robot in the Atacama Desert, and found the behaviour consistent with the simulation results. The samples collected by the algorithm supported the hypothesis favoured by the planetary scientists in the field.

This work demonstrates that robotic scientific explorers can plan actions that inform the belief in different hypotheses, and do so without reliance on deep knowledge of the hypotheses themselves. This, in turn sets a new approach to science autonomy which is not focused exclusively on collecting data but to let robots explore on the behalf of scientists collecting data relevant to the scientists’ hypotheses. Further, the collected data could be used to support hypotheses generation techniques like those described in section 2.4.

Collecting data in the context of hypotheses returns to the notion of scientific as a cyclic process. We have demonstrated how robots can use multiple competing hypotheses to determine what new experiments ought to be conducted, but what remains is the question of when and how to generate new hypotheses. This work is the first to direct robot action selection in order to falsify hypotheses, and is a necessary component for a robot that is simultaneously generating and testing hypotheses.

6.2 Limitations

As stated previously, empirical work can naturally only speak to the circumstances under which the data were collected. While our experiments demonstrate statistically significant improvements in performance for the proposed algorithm, hypotheses about performance outside of the stated conditions must be rigorously supported with experimentation.

**Foraging** The foraging algorithm makes the assumption that the next encountered object is drawn from a distribution independently of any previously encountered object. Modelling the next object to arrive with a Markov chain would further improve the fidelity of the algorithm to field operations, and should decay gracefully when the encountered classes of objects are independently and identically distributed.

The foraging system as described in this document does not consider the mission objectives beyond the cost for sampling and exploring, and even then it employs a greedy approach. It would be beneficial to integrate the foraging algorithm with a higher-level scheduler like CASPER (Knight et al., 2001), in order to prioritize opportunistic sampling in the context of mission objectives.

**Prospecting** The prospecting algorithm that we presented in this document intentionally ignored the fact that the one dimensional signal the algorithm was monitoring was embedded in a two dimensional environment. There are almost certainly advantages to be gained by recognizing that the robot is operating in a more complex environment. Like foraging it is possible that
the requests to deploy an AIM could be scheduled by a higher-level system that could account for the spacing between AIMS, and the degree of overlap from previous deployments.

Falsification Our approach to falsification only considered a small number of hypotheses, testing with a larger pool of hypotheses is warranted. While adding more hypotheses should not reduce the ability to identify which of the hypotheses are most likely to be correct, it could have deleterious effects on the ability to recognize that none of the hypotheses are accurate, in the case that the algorithm has been assigned a pool of poor hypotheses.

Further, this algorithm does not update hypotheses in response to the data that have been collected. Being able to revise hypotheses in the light of new evidence would increase the ability of the robot to learn the correct hypothesis. However, for design purposes it would be keep the knowledge of how to update the hypotheses separate from the action planning algorithm.

Samples were selected such that they maximized the sum of the information gained in distribution over samples collected and the information gained in the belief over the hypotheses. It is possible that the samples collected contribute moderate information gains for either component of the sum, and do not yield considerable information in either the domain of the hypotheses or the belief in the hypotheses. It is possible that other value assignment functions are required.

There is lacking a measure of salience of the results. In the algorithm as it is designed no one observation can disqualify a hypothesis. In fact, if a considerable amount of favourable data have been collected, it could conceivably prove challenging to unseat a hypothesis. Partially this is a function of the codomain of the hypotheses being relatively small, but the algorithm should be modified to account for this.

One of the stated objectives of the algorithm was to identify when a new hypothesis is needed, by maintaining a belief in a hypothetical hypothesis, \( H_0 \). It might be that the variance across the belief in the hypotheses is a better indicator that a new hypothesis is needed. Regardless, it remains a problem to determine when to search for a new hypothesis and when to sample amongst the existing hypotheses.

6.3 Future Work

While the work in this thesis advances science autonomy algorithms, there still remains work to do. To that end we identify a number of extensions for the algorithms presented in this document, and to science autonomy in general, that should be undertaken to improve the state of the art.

In the presented work we used a measure of information gain based on Shannon’s definition of entropy. As discussed in Chapter 2, there are advantages to using the Cauchy-Schwartz Quadratic Mutual Information (Principe, 2010) in information gathering algorithms, chiefly the increased speed of computation. The performance of the algorithms presented in this thesis could be improved by using the Cauchy-Schwartz Quadratic Mutual Information criterion. The Cauchy-Schwartz formulation symbolically

Foraging The algorithm should be modified to recognize that the arrival distribution of the different classes of objects is not memoryless. This would make the algorithm more amenable
to operations in environments where there the next sampling opportunity is highly dependent on the currently available option.

The authors would like to investigate different cost/benefit analyses for the exploration vs sampling decision. As noted previously, using the ratio of reward to cost is potentially numerically unstable, therefore different strategies should be investigated.

The algorithm tested in this document only considered objects where the underlying distribution was Bernoulli. Other distributions should be tested in order to be more reflective of the variety of observations – real valued, vector valued – that can be collected by robots in the field.

The algorithm should be integrated with a mission scheduler and tested on a real robot. During deployment it would also be worthwhile to integrate the algorithm with some form of classifier that identifies targets of interest in the environment. Consequently the algorithm will need to be updated to account for confusion between the different classes of objects.

**Prospecting** The prospecting algorithm should be extended with a planner that makes more rigorous decisions about resource trade-offs and mission risks incurred in more involved sampling processes like drilling. The addition of confidence in change detection should let one apply traditional decision-making rules without human supervision.

Multi-hypotheses testing would let a robot scientist deal with a broader range of scenarios while exploring. Multi-hypothesis testing would also let the algorithm consider more than one number of change points in the underlying distribution at any given time.

The AIMs that were deployed by the prospecting algorithm were simple Archimedean spirals. It would be useful to deploy AIMs that were more informed by data previously collected. This would be an excellent application of an algorithm like that deployed by Wilson and Williams (2017). A more sophisticated plan for AIM deployment could account for higher dimensional algorithm that the change detection algorithm ignored.

**Falsification** The falsification planning algorithm needs to incorporate uncertainty in prediction and also noisy sensors. However, this is a straightforward application of Bayes’ chain rule. But perhaps the most important question that needs to be addressed as the next step is when is it necessary to create a new hypothesis. If we use the quantity $P(H_0)$ to determine if a new hypothesis is required it is possible that early observations could cause an excessive amount of time spent in generating new hypotheses before a sufficiently informative dataset is collected. As discussed in chapter 5, the automatic determination of the number of samples required, under a PAC learning framework, to be confident in the predictions of a hypothesis would be immensely valuable. This is a necessary step forward in simultaneous exploration and hypothesis generation.

**Science Autonomy, in general** The perception algorithms that are used in science autonomy focus around fairly basic classifiers, and most of them are pre-trained before they are deployed. Having a perception system that learns in situ could help distinguish a wider range of scientifically interesting phenomena. DEMUD Wagstaff et al. (2013) is an important first step in this approach. Given the progress being made in artificial neural networks, they would be a natural avenue to explore. However the training of neural networks relies of large volumes of data and powerful computers. The computational demands are antithetical to the computing resources
available in flight missions. On the one hand, one could find a middle ground between basic, pre-trained classifiers and full deep learning system, such as extreme learning machines [Huang et al. (2011)]. On the other hand, one could explore alternative computing architectures.

Non-traditional computer architectures, as discussed in [Younger et al., 2014; Traversa et al., 2015], are an area that should be investigated for building robot scientists. Traversa et al. (2015) claim their computing hardware is able to solve non-polynomial problems in polynomial time. Given that solving information gathering problems can be computationally intensive, it is worth evaluating whether these new kinds of computing could improve the performance of robot scientist and what, if any, effect their use would have on scientific decision making processes.

An other interesting advanced is the development of language for individual robots exploring both here on Earth and in the universe. This language must be grounded in sensory apparatus and also be temporally aware. A lot of information is compressed in language, and humans used metaphors and analogies to do reasoning. Imbuing a robot scientist with some of that ability could help speed up hypothesis formation. This is again something that can be addressed with non-traditional computing [Kanerva et al., 2000].
Appendices
Appendix A

List of Terms

**Entropy** Assumed to be Shannon’s Entropy, $H(p)$ of a distribution $p(x)$, defined to be $H(p) = -\sum_{x \in X} p(x) \log_2 p(x)$.

**Hypothesis** A function that predicts from an input space $X$ to a probability distribution over an observation space, $Z$. Formally: $h : X \rightarrow (Z \rightarrow [0, 1])$

**I.I.D.** Identically and independently distributed.

**Object** An entity in the environment the robot can reason about.

**Opportunistic Sampling** Sampling conducted in response to sampling opportunities that were not or could not be anticipated.

**PAC Learning** Probably Approximately Correct Learning, invented by Leslie Valiant. A branch of computational learning theory that specialises in putting confidence bounds on learned functions.

**Perplexity** A measure of how well a probability model describes an observation.

**Primary or Proxy Sensor** This is the instrument that is used to identify sampling opportunities. For example, a camera which can be used to determine the type of rock visible.

**Prospecting** The navigation of a scalar or vector field in order to seek locations that maximize an objective function.

**Prospecting Sensor** The primary sensor used during prospecting.

**Sampling Resources** Material that is consumed during the sampling process. For example, chemical reagents or sample containers.

**Sampling Opportunity** An object that the robot can choose to sample with one of its instruments, for some cost.

**Secondary Sensor** This is the instrument that is used to extract observations from sampling opportunities identified by the primary sensor. Secondary sensors can be more expensive in time, energy, or sampling resources than the primary sensor.

**Speed Made Good** The speed of a vehicle towards the goal. Less than the speed of the vehicle unless the vehicle is headed directly towards the goal.

**Surprise** As defined by Claude Shannon, surprise of an event $x$ given a probability distribution

147
\[ p \text{ is } - \log p(x) \]

**Transect** A path along which one collects data.
Appendix B

Opportunistic Sampling of Discrete Objects (Foraging) Supplemental Material

In this appendix we present the performance of the competing algorithms (Foraging, Uniform, and Greedy) against a sampling algorithm which always engages with the environment, called “Always Engage”.

B.1 Experiment 1 - Uniform Arrival Distribution, Different Underlying Distributions

In the case where all objects are equally likely the Foraging algorithm has the least bad performance relative to the always engage algorithm.

Figure B.1: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.
Figure B.2: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.

Figure B.3: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.

Notice that the Foraging algorithm has the least amount of underperformance relative to Always Engage over the space of sampling and exploration costs.

B.2 Experiment 2 - Skewed Arrival Distribution with Identical Underlying Distributions

Here we give the performance of the competing algorithms with respect to the Always Engage algorithm with the unbalanced arrival distribution. These distributions test performance of the algorithms when $K = 6$ objects.
Figure B.4: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with $P(X)$ being much larger than for other classes of objects. $P(Z|X = x) = 0.3 \forall x \in X$.

Figure B.5: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with $P(X)$ being much larger than for other classes of objects. $P(Z|X = x) = 0.3 \forall x \in X$. 

$P(X = 1) = 0.9, N=50$ $P(X = 1) = 0.8, N=50$ $P(X = 1) = 0.7, N=50$ $P(X = 1) = 0.6, N=50$ $P(X = 1) = 0.5, N=50$
Figure B.6: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with \( P(X) \) being much larger than for other classes of objects. \( P(Z|X=x) = 0.3 \forall x \in X \).

Uniform (Figure B.4) and Greedy (Figure B.5) perform as good as or better than Always Engage for most settings of exploration and sampling costs. The Foraging algorithm never underperforms the Always Engage algorithm (Figure B.6).

Next we show the performance of the competing algorithms with respect to the Always Engage algorithm with a Zipfian arrival distribution, where \( s = 1 \) and \( K \in \{5, 6, 7, 8\} \). Again, the Foraging algorithm never underperforms the Always Engage algorithm (Figure B.9).

Figure B.7: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with \( s = 1 \) and \( K \in \{5, 6, 7, 8\} \).
Figure B.8: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with \( s = 1 \) and \( K \in \{5, 6, 7, 8\} \).

Figure B.9: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with \( s = 1 \) and \( K \in \{5, 6, 7, 8\} \).

B.3 Experiment 3 - Skewed Arrival Distribution with Distractor Object

These plots show the performance of the competing algorithms with respect to the Always Engage algorithm first with the unbalanced arrival distribution with \( P(X) = 0.8 \) and \( K = 8 \), and next with a Zipfian distribution with \( s = 1, K = 8 \).

Uniform’s underperforming of the Always Engage algorithm is more pronounced when a Zipfian distribution is followed (Figure B.10). There appears to be no difference in performance
as a function of the underlying distribution of the modified object class. This also holds for the Greedy algorithm (Figure B.11). Again, the Foraging algorithm never underperforms the Always Engage algorithm (Figure B.12).

\[ P(Z|X = 2) = 0.1, \text{N}=50 \quad P(Z|X = 2) = 0.3 \text{ N}=50 \quad P(Z|X = 2) = 0.5 \text{ N}=50 \quad P(Z|X = 2) = 0.6, \text{ N}=50 \]

\[ \Delta D_{KL}(\%) \]

\[ \text{Zipf}(s = 1, K = 8), \text{ N}=50 \quad \text{Zipf}(s = 1, K = 8), \text{ N}=50 \quad \text{Zipf}(s = 1, K = 8), \text{ N}=50 \quad \text{Zipf}(s = 1, K = 8), \text{ N}=50 \]

**Figure B.10:** Uniform sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.
$P(Z|X = 2) = 0.1$, N=50  
$P(Z|X = 2) = 0.3$, N=50  
$P(Z|X = 2) = 0.5$, N=50  
$P(Z|X = 2) = 0.6$, N=50

Figure B.11: Greedy sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.
Figure B.12: Foraging sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

B.4 Experiment 4 - Skewed Arrival Distribution with Random Underlying Distributions

These plots show the performance of the competing algorithms with respect to the Always Engage algorithm first with the unbalanced arrival distribution with $P(X) = 0.8$ and $K = 8$, and next with a Zipfian distribution with $s = 1, K = 8$.

Uniform sampling performs better than Always Engage for most sampling costs, with a degradation that is much more pronounced when the arrival distribution follows Zipf’s law (Figure B.13). Notice that the Uniform sampling performs worse for RAND3 than for the other settings of the underlying distributions.

The Greedy algorithm (Figure B.14) appears fairly robust to the changes in the underlying
distribution. Generally good performance for small exploration costs, excepting small sampling cost, where the Always Engage algorithm performs better.

The Foraging algorithm also has performance that is sensitive to the settings of the underlying distributions, as can be seen in Figure B.15. Notice only in one setting, the RAND3 settings for the underlying distribution and the unbalanced arrival distribution, does the Foraging algorithm perform worse than the Always Engage algorithm. The worse performance is localized to large settings of exploration cost and sampling cost.

**Figure B.13:** Uniform sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.
Figure B.14: Greedy sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.
Figure B.15: Foraging sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.
Appendix C

Opportunistic Sampling in a Scalar Field (Prospecting) Supplemental Material

C.1 Experiment 1 - Change in Underlying Distribution Rate

The following tables show the effect size comparing the SPRT algorithm to the Memory Threshold algorithm.

Table C.1: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.

<table>
<thead>
<tr>
<th>( \lambda_2 - \lambda_1 )</th>
<th>( \mu[FN]<em>{Mem\text{Thresh}} - \mu[FN]</em>{SPRT(s=2)} )</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>1.98 \times 10^{3}</td>
</tr>
<tr>
<td>-20</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>1.98 \times 10^{3}</td>
</tr>
<tr>
<td>-10</td>
<td>0.99</td>
<td>[0.979, 0.993]</td>
<td>68.8</td>
</tr>
<tr>
<td>0</td>
<td>0.49</td>
<td>[0.380, 0.589]</td>
<td>1.86</td>
</tr>
<tr>
<td>10</td>
<td>0.98</td>
<td>[0.971, 0.988]</td>
<td>49.6</td>
</tr>
<tr>
<td>20</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>1.98 \times 10^{3}</td>
</tr>
<tr>
<td>30</td>
<td>0.88</td>
<td>[0.867, 0.897]</td>
<td>29.6</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.460, 0.500]</td>
<td>9.79</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.139]</td>
<td>3.51</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.016, 0.027]</td>
<td>1.51</td>
</tr>
</tbody>
</table>

The following tables show the effect size on the mean error rate in predicting the time of change as a function of the change in the underlying rate.
Table C.2: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(s=4)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>$1.98 \times 10^3$</td>
</tr>
<tr>
<td>-20</td>
<td>1.00</td>
<td>[0.995, 1]</td>
<td>$1.96 \times 10^3$</td>
</tr>
<tr>
<td>-10</td>
<td>0.95</td>
<td>[0.932, 0.958]</td>
<td>30.20</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>0.02</td>
</tr>
<tr>
<td>10</td>
<td>0.95</td>
<td>[0.934, 0.953]</td>
<td>32.050</td>
</tr>
<tr>
<td>20</td>
<td>0.99</td>
<td>[0.986, 1.000]</td>
<td>133.00</td>
</tr>
<tr>
<td>30</td>
<td>0.88</td>
<td>[0.868, 0.892]</td>
<td>36.70</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.462, 0.500]</td>
<td>9.81</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.139]</td>
<td>3.53</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.015, 0.027]</td>
<td>1.54</td>
</tr>
</tbody>
</table>

Table C.3: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(s=8)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1.000, 1.000]</td>
<td>$1.98 \times 10^3$</td>
</tr>
<tr>
<td>-20</td>
<td>0.98</td>
<td>[0.975, 0.983]</td>
<td>99.30</td>
</tr>
<tr>
<td>-10</td>
<td>0.87</td>
<td>[0.862, 0.884]</td>
<td>16.10</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>0.87</td>
<td>[0.844, 0.903]</td>
<td>13.50</td>
</tr>
<tr>
<td>20</td>
<td>0.98</td>
<td>[0.968, 0.981]</td>
<td>96.80</td>
</tr>
<tr>
<td>30</td>
<td>0.87</td>
<td>[0.858, 0.882]</td>
<td>28.60</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.459, 0.504]</td>
<td>11.00</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.138]</td>
<td>3.55</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.015, 0.027]</td>
<td>1.54</td>
</tr>
</tbody>
</table>

The follow plots show the size of the effect on the false positive rate as a function of the change in the underlying rate.
Table C.4: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2. Here the SPRT 2 algorithm is demonstrably worse in terms of false positive rate than the memory threshold algorithm.

<table>
<thead>
<tr>
<th>Rate Change</th>
<th>(\mu_{FP}^{MemThresh} - \mu_{FP}^{SPRT(s=2)})</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.61</td>
</tr>
<tr>
<td>-20</td>
<td>-0.25</td>
<td>[-0.313, -0.176]</td>
<td>-1.65</td>
</tr>
<tr>
<td>-10</td>
<td>-0.25</td>
<td>[-0.314, -0.176]</td>
<td>-1.66</td>
</tr>
<tr>
<td>0</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.60</td>
</tr>
<tr>
<td>10</td>
<td>-0.25</td>
<td>[-0.313, -0.177]</td>
<td>-1.67</td>
</tr>
<tr>
<td>20</td>
<td>-0.25</td>
<td>[-0.317, -0.177]</td>
<td>-1.64</td>
</tr>
<tr>
<td>30</td>
<td>-0.25</td>
<td>[-0.314, -0.177]</td>
<td>-1.60</td>
</tr>
<tr>
<td>40</td>
<td>-0.25</td>
<td>[-0.314, -0.178]</td>
<td>-1.67</td>
</tr>
<tr>
<td>50</td>
<td>-0.25</td>
<td>[-0.317, -0.180]</td>
<td>-1.67</td>
</tr>
<tr>
<td>60</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.62</td>
</tr>
</tbody>
</table>

Table C.5: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4. The performance of the two algorithms is indistinguishable at a 95% confidence level.

<table>
<thead>
<tr>
<th>Rate Change</th>
<th>(\mu_{FP}^{MemThresh} - \mu_{FP}^{SPRT(s=4)})</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>0.00</td>
<td>[0.014, 0.0001]</td>
<td>N/A</td>
</tr>
<tr>
<td>-20</td>
<td>0.00</td>
<td>[-0.016, 0.0001]</td>
<td>N/A</td>
</tr>
<tr>
<td>-10</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>N/A</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>[-0.010, 0.0001]</td>
<td>N/A</td>
</tr>
<tr>
<td>20</td>
<td>-0.01</td>
<td>[-0.018, 0.0001]</td>
<td>N/A</td>
</tr>
<tr>
<td>30</td>
<td>-0.01</td>
<td>[-0.018, 0.00003]</td>
<td>N/A</td>
</tr>
<tr>
<td>40</td>
<td>-0.002</td>
<td>[-0.015, 0.0002]</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>-0.001</td>
<td>[-0.009, 0.0002]</td>
<td>N/A</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table C.6: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8. The two algorithms are indistinguishable at a 95% confidence level.

<table>
<thead>
<tr>
<th>Rate Change</th>
<th>$\mu[FP]<em>{\text{MemThresh}} - \mu[FP]</em>{\text{SPRT}(s=8)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>-20</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>-10</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>30</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>40</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table C.7: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2. The two algorithms are indistinguishable at a 95% confidence level.

<table>
<thead>
<tr>
<th>Rate Change</th>
<th>$\mu[FP]<em>{\text{MemThresh}} - \mu[FP]</em>{\text{SPRT}(s=2)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>44.15</td>
<td>[45.683, 42.617]</td>
<td>56.44</td>
</tr>
<tr>
<td>-20</td>
<td>44.04</td>
<td>[45.569, 42.508]</td>
<td>56.41</td>
</tr>
<tr>
<td>-10</td>
<td>42.77</td>
<td>[44.343, 41.195]</td>
<td>53.26</td>
</tr>
<tr>
<td>0</td>
<td>24.08</td>
<td>[27.045, 21.106]</td>
<td>15.89</td>
</tr>
<tr>
<td>10</td>
<td>42.95</td>
<td>[44.501, 41.407]</td>
<td>54.42</td>
</tr>
<tr>
<td>20</td>
<td>35.48</td>
<td>[40.028, 30.927]</td>
<td>15.28</td>
</tr>
<tr>
<td>30</td>
<td>18.80</td>
<td>[21.248, 16.357]</td>
<td>15.07</td>
</tr>
<tr>
<td>40</td>
<td>18.74</td>
<td>[20.391, 17.086]</td>
<td>22.22</td>
</tr>
<tr>
<td>60</td>
<td>2.11</td>
<td>[−0.150, 4.365]</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table C.8: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4. The two algorithms are indistinguishable at a 95% confidence level.

<table>
<thead>
<tr>
<th>Rate Change $\lambda_2 - \lambda_1$</th>
<th>$\mu[FP]<em>{MemThresh} - \mu[FP]</em>{SPRT(s=4)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>49.62</td>
<td>[49.834, 49.409]</td>
<td>457.27</td>
</tr>
<tr>
<td>-20</td>
<td>49.50</td>
<td>[49.723, 49.285]</td>
<td>443.31</td>
</tr>
<tr>
<td>-10</td>
<td>48.20</td>
<td>[48.759, 47.648]</td>
<td>170.17</td>
</tr>
<tr>
<td>0</td>
<td>18.06</td>
<td>[22.923, 13.205]</td>
<td>7.29</td>
</tr>
<tr>
<td>10</td>
<td>48.41</td>
<td>[48.850, 47.962]</td>
<td>213.69</td>
</tr>
<tr>
<td>20</td>
<td>40.96</td>
<td>[45.243, 36.671]</td>
<td>18.73</td>
</tr>
<tr>
<td>40</td>
<td>24.21</td>
<td>[24.850, 23.562]</td>
<td>73.69</td>
</tr>
<tr>
<td>50</td>
<td>18.60</td>
<td>[19.598, 17.594]</td>
<td>36.36</td>
</tr>
<tr>
<td>60</td>
<td>7.58</td>
<td>[9.249, 5.908]</td>
<td>8.89</td>
</tr>
</tbody>
</table>

Table C.9: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8. The two algorithms are indistinguishable at a 95% confidence level.

<table>
<thead>
<tr>
<th>Rate Change $\lambda_2 - \lambda_1$</th>
<th>$\mu[FP]<em>{MemThresh} - \mu[FP]</em>{SPRT(s=8)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>50.00</td>
<td>[50.0, 50.0]</td>
<td>inf</td>
</tr>
<tr>
<td>-20</td>
<td>49.89</td>
<td>[49.975, 49.807]</td>
<td>1167.42</td>
</tr>
<tr>
<td>-10</td>
<td>48.62</td>
<td>[49.145, 48.088]</td>
<td>180.34</td>
</tr>
<tr>
<td>0</td>
<td>0.82</td>
<td>[−0.771, 2.411]</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>48.89</td>
<td>[49.266, 48.508]</td>
<td>252.90</td>
</tr>
<tr>
<td>20</td>
<td>41.34</td>
<td>[45.617, 37.055]</td>
<td>18.93</td>
</tr>
<tr>
<td>30</td>
<td>24.64</td>
<td>[26.544, 22.735]</td>
<td>25.35</td>
</tr>
<tr>
<td>40</td>
<td>24.58</td>
<td>[25.191, 23.977]</td>
<td>79.33</td>
</tr>
<tr>
<td>50</td>
<td>18.97</td>
<td>[19.953, 17.995]</td>
<td>37.97</td>
</tr>
</tbody>
</table>
C.2 Experiment 2 - Effect of Delay of Change Onset

Table C.10: The average reduction in false positive rate due to using the SPRT algorithm vs the Memory Threshold algorithm.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>$\mu[FP]<em>{\text{MemThresh}} - \mu[FP]</em>{\text{sprt}}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.016</td>
<td>[0.009, 0.022]</td>
<td>0.96</td>
</tr>
<tr>
<td>80</td>
<td>0.016</td>
<td>[0.011, 0.020]</td>
<td>1.96</td>
</tr>
<tr>
<td>120</td>
<td>0.017</td>
<td>[0.014, 0.020]</td>
<td>2.84</td>
</tr>
<tr>
<td>160</td>
<td>0.016</td>
<td>[0.013, 0.018]</td>
<td>3.13</td>
</tr>
<tr>
<td>200</td>
<td>0.016</td>
<td>[0.014, 0.018]</td>
<td>5.08</td>
</tr>
<tr>
<td>240</td>
<td>0.017</td>
<td>[0.015, 0.020]</td>
<td>4.72</td>
</tr>
<tr>
<td>260</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>4.61</td>
</tr>
<tr>
<td>320</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>4.94</td>
</tr>
<tr>
<td>360</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>5.72</td>
</tr>
</tbody>
</table>

Table C.11: The reduction in FN rate from the Memory Threshold to the SPRT algorithm.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>$\mu[FN]<em>{\text{MemThresh}} - \mu[FN]</em>{\text{sprt}}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.029</td>
<td>[0.027, 0.031]</td>
<td>5.14</td>
</tr>
<tr>
<td>80</td>
<td>0.029</td>
<td>[0.026, 0.031]</td>
<td>4.78</td>
</tr>
<tr>
<td>120</td>
<td>0.028</td>
<td>[0.026, 0.031]</td>
<td>4.41</td>
</tr>
<tr>
<td>160</td>
<td>0.028</td>
<td>[0.025, 0.031]</td>
<td>4.18</td>
</tr>
<tr>
<td>200</td>
<td>0.027</td>
<td>[0.025, 0.030]</td>
<td>6.19</td>
</tr>
<tr>
<td>240</td>
<td>0.026</td>
<td>[0.022, 0.030]</td>
<td>2.65</td>
</tr>
<tr>
<td>260</td>
<td>0.025</td>
<td>[0.021, 0.030]</td>
<td>2.52</td>
</tr>
<tr>
<td>320</td>
<td>0.025</td>
<td>[0.019, 0.031]</td>
<td>1.57</td>
</tr>
<tr>
<td>360</td>
<td>0.018</td>
<td>[0.008, 0.027]</td>
<td>0.73</td>
</tr>
</tbody>
</table>
Table C.12: The effect size of the reduction in mean error as the change point is varied.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>$\mu[MeanError]<em>{MemThresh} - \mu[MeanError]</em>{sprt}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>155</td>
<td>[155, 157]</td>
<td>39.3</td>
</tr>
<tr>
<td>80</td>
<td>136</td>
<td>[134, 138]</td>
<td>44.4</td>
</tr>
<tr>
<td>120</td>
<td>115</td>
<td>[112, 117]</td>
<td>25.1</td>
</tr>
<tr>
<td>160</td>
<td>94.2</td>
<td>[91.6, 96.9]</td>
<td>19.1</td>
</tr>
<tr>
<td>200</td>
<td>73.9</td>
<td>[70.7, 77]</td>
<td>13.8</td>
</tr>
<tr>
<td>240</td>
<td>51.9</td>
<td>[49.2, 54.5]</td>
<td>10.4</td>
</tr>
<tr>
<td>260</td>
<td>42.6</td>
<td>[40.2, 45]</td>
<td>9.47</td>
</tr>
<tr>
<td>320</td>
<td>11.8</td>
<td>[9.35, 14.2]</td>
<td>2.81</td>
</tr>
<tr>
<td>360</td>
<td>16.1</td>
<td>[−18.9, −13.4]</td>
<td>-3.42</td>
</tr>
</tbody>
</table>

C.3  Experiment 4 - Effect on Performance in 2D Operations

Here we report the number of maxima in the underlying maps which were observed by the robot using the different algorithms, as well as the number of AIMs deployed. For each map 10 trials were run, and we report the average number and standard deviation for the different quantities observed.
Table C.13: The average number of peaks captured by the three different algorithms on each map. We report average ± one standard deviation, \( n = 10 \).

<table>
<thead>
<tr>
<th>Map Number</th>
<th>No AIMs</th>
<th>Memory Threshold</th>
<th>Adaptive Threshold</th>
<th>Relative Change</th>
<th>SPRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>10.0 ± 0.0</td>
<td>16.6 ± 0.3</td>
<td>47.4 ± 4.1</td>
<td>88.2 ± 4.0</td>
<td>42.9 ± 3.1</td>
</tr>
<tr>
<td>01</td>
<td>11.0 ± 0.0</td>
<td>11.0 ± 0.0</td>
<td>44.8 ± 3.4</td>
<td>87.4 ± 3.8</td>
<td>44.1 ± 2.5</td>
</tr>
<tr>
<td>02</td>
<td>20.0 ± 0.0</td>
<td>21.0 ± 0.0</td>
<td>48.9 ± 2.9</td>
<td>95.5 ± 2.9</td>
<td>53.5 ± 4.8</td>
</tr>
<tr>
<td>03</td>
<td>13.0 ± 0.0</td>
<td>16.6 ± 0.3</td>
<td>49.0 ± 2.5</td>
<td>95.1 ± 1.9</td>
<td>46.3 ± 3.3</td>
</tr>
<tr>
<td>04</td>
<td>15.0 ± 0.0</td>
<td>16.1 ± 0.7</td>
<td>50.5 ± 3.7</td>
<td>96.9 ± 2.9</td>
<td>54.1 ± 2.5</td>
</tr>
<tr>
<td>05</td>
<td>12.0 ± 0.0</td>
<td>13.2 ± 1.0</td>
<td>50.2 ± 7.1</td>
<td>90.4 ± 4.0</td>
<td>53.1 ± 3.0</td>
</tr>
<tr>
<td>06</td>
<td>19.0 ± 0.0</td>
<td>19.2 ± 0.4</td>
<td>49.0 ± 5.8</td>
<td>88.6 ± 3.3</td>
<td>56.9 ± 1.9</td>
</tr>
<tr>
<td>07</td>
<td>16.0 ± 0.0</td>
<td>17.4 ± 0.4</td>
<td>43.8 ± 5.2</td>
<td>88.9 ± 3.1</td>
<td>49.9 ± 3.0</td>
</tr>
<tr>
<td>08</td>
<td>12.0 ± 0.0</td>
<td>15.9 ± 0.4</td>
<td>44.7 ± 5.3</td>
<td>89.7 ± 3.2</td>
<td>48.3 ± 2.5</td>
</tr>
<tr>
<td>09</td>
<td>17.0 ± 0.0</td>
<td>18.2 ± 0.4</td>
<td>47.0 ± 4.4</td>
<td>92.3 ± 3.5</td>
<td>53.0 ± 2.9</td>
</tr>
<tr>
<td>10</td>
<td>12.0 ± 0.0</td>
<td>12.4 ± 0.4</td>
<td>43.1 ± 5.8</td>
<td>93.9 ± 2.6</td>
<td>51.0 ± 3.6</td>
</tr>
<tr>
<td>11</td>
<td>15.0 ± 0.0</td>
<td>15.1 ± 0.2</td>
<td>47.3 ± 3.2</td>
<td>93.7 ± 4.8</td>
<td>48.7 ± 2.0</td>
</tr>
<tr>
<td>12</td>
<td>18.0 ± 0.0</td>
<td>19.4 ± 0.4</td>
<td>51.1 ± 6.7</td>
<td>100.8 ± 2.6</td>
<td>53.0 ± 2.7</td>
</tr>
<tr>
<td>13</td>
<td>19.0 ± 0.0</td>
<td>19.0 ± 0.0</td>
<td>52.1 ± 3.1</td>
<td>93.1 ± 2.3</td>
<td>52.0 ± 1.9</td>
</tr>
<tr>
<td>14</td>
<td>24.0 ± 0.0</td>
<td>24.1 ± 0.2</td>
<td>51.0 ± 4.2</td>
<td>96.9 ± 2.5</td>
<td>56.9 ± 2.7</td>
</tr>
<tr>
<td>15</td>
<td>17.0 ± 0.0</td>
<td>19.1 ± 0.7</td>
<td>52.8 ± 3.3</td>
<td>97.5 ± 3.2</td>
<td>53.4 ± 4.5</td>
</tr>
<tr>
<td>16</td>
<td>19.0 ± 0.0</td>
<td>24.2 ± 0.4</td>
<td>50.1 ± 5.9</td>
<td>100.2 ± 4.5</td>
<td>59.9 ± 3.4</td>
</tr>
<tr>
<td>17</td>
<td>16.0 ± 0.0</td>
<td>16.0 ± 0.0</td>
<td>53.1 ± 5.1</td>
<td>97.2 ± 3.8</td>
<td>54.4 ± 2.7</td>
</tr>
<tr>
<td>18</td>
<td>18.0 ± 0.0</td>
<td>21.1 ± 1.2</td>
<td>48.5 ± 3.3</td>
<td>88.1 ± 3.2</td>
<td>50.3 ± 3.1</td>
</tr>
<tr>
<td>19</td>
<td>10.0 ± 0.0</td>
<td>12.3 ± 0.9</td>
<td>38.2 ± 3.6</td>
<td>92.3 ± 2.9</td>
<td>46.9 ± 3.2</td>
</tr>
<tr>
<td>20</td>
<td>22.0 ± 0.0</td>
<td>23.4 ± 0.4</td>
<td>52.5 ± 4.3</td>
<td>93.9 ± 2.1</td>
<td>56.4 ± 2.5</td>
</tr>
<tr>
<td>21</td>
<td>20.0 ± 0.0</td>
<td>21.6 ± 0.6</td>
<td>57.3 ± 5.4</td>
<td>95.1 ± 2.1</td>
<td>51.6 ± 2.3</td>
</tr>
<tr>
<td>22</td>
<td>19.0 ± 0.0</td>
<td>19.5 ± 0.3</td>
<td>55.0 ± 5.6</td>
<td>108.9 ± 3.2</td>
<td>63.5 ± 2.5</td>
</tr>
<tr>
<td>23</td>
<td>16.0 ± 0.0</td>
<td>16.4 ± 0.4</td>
<td>45.3 ± 3.4</td>
<td>89.4 ± 3.7</td>
<td>51.5 ± 2.4</td>
</tr>
<tr>
<td>24</td>
<td>18.0 ± 0.0</td>
<td>20.5 ± 0.7</td>
<td>47.8 ± 3.0</td>
<td>91.1 ± 3.5</td>
<td>56.3 ± 3.8</td>
</tr>
<tr>
<td>25</td>
<td>13.0 ± 0.0</td>
<td>15.8 ± 0.4</td>
<td>43.7 ± 3.9</td>
<td>80.2 ± 3.8</td>
<td>42.0 ± 1.4</td>
</tr>
<tr>
<td>26</td>
<td>14.0 ± 0.0</td>
<td>16.2 ± 0.4</td>
<td>49.9 ± 5.7</td>
<td>89.8 ± 2.1</td>
<td>49.4 ± 3.5</td>
</tr>
<tr>
<td>27</td>
<td>19.0 ± 0.0</td>
<td>20.6 ± 0.6</td>
<td>49.0 ± 2.5</td>
<td>94.3 ± 5.2</td>
<td>53.6 ± 3.2</td>
</tr>
<tr>
<td>28</td>
<td>22.0 ± 0.0</td>
<td>24.0 ± 0.5</td>
<td>52.5 ± 5.5</td>
<td>97.3 ± 4.1</td>
<td>55.8 ± 3.5</td>
</tr>
<tr>
<td>29</td>
<td>12.0 ± 0.0</td>
<td>12.9 ± 0.7</td>
<td>45.0 ± 3.5</td>
<td>86.0 ± 2.4</td>
<td>52.8 ± 2.7</td>
</tr>
</tbody>
</table>
Table C.14: The average number of AIMS deployed by the different algorithms on each map. We report average ± one standard deviation, \( n = 10 \)

<table>
<thead>
<tr>
<th>Map Number</th>
<th>No AIMS</th>
<th>Memory Threshold</th>
<th>Adaptive Threshold</th>
<th>Relative Change</th>
<th>SPRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.0 ± 0.0</td>
<td>3.3 ± 0.3</td>
<td>60.9 ± 3.1</td>
<td>109.0 ± 1.7</td>
<td>39.9 ± 1.9</td>
</tr>
<tr>
<td>01</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>53.2 ± 1.9</td>
<td>107.5 ± 1.7</td>
<td>39.8 ± 0.7</td>
</tr>
<tr>
<td>02</td>
<td>0.0 ± 0.0</td>
<td>1.1 ± 0.2</td>
<td>56.5 ± 2.2</td>
<td>108.1 ± 0.9</td>
<td>43.8 ± 1.3</td>
</tr>
<tr>
<td>03</td>
<td>0.0 ± 0.0</td>
<td>5.4 ± 0.6</td>
<td>56.0 ± 1.8</td>
<td>109.8 ± 2.2</td>
<td>37.3 ± 1.4</td>
</tr>
<tr>
<td>04</td>
<td>0.0 ± 0.0</td>
<td>2.6 ± 0.4</td>
<td>56.2 ± 2.3</td>
<td>109.6 ± 1.1</td>
<td>41.9 ± 1.0</td>
</tr>
<tr>
<td>05</td>
<td>0.0 ± 0.0</td>
<td>1.0 ± 0.4</td>
<td>53.1 ± 2.7</td>
<td>108.8 ± 0.7</td>
<td>43.2 ± 1.5</td>
</tr>
<tr>
<td>06</td>
<td>0.0 ± 0.0</td>
<td>1.2 ± 0.4</td>
<td>52.2 ± 1.8</td>
<td>108.4 ± 1.5</td>
<td>38.2 ± 0.9</td>
</tr>
<tr>
<td>07</td>
<td>0.0 ± 0.0</td>
<td>2.0 ± 0.0</td>
<td>53.9 ± 1.9</td>
<td>107.6 ± 1.6</td>
<td>43.6 ± 2.3</td>
</tr>
<tr>
<td>08</td>
<td>0.0 ± 0.0</td>
<td>5.1 ± 0.2</td>
<td>54.1 ± 2.4</td>
<td>108.8 ± 2.1</td>
<td>42.0 ± 1.4</td>
</tr>
<tr>
<td>09</td>
<td>0.0 ± 0.0</td>
<td>2.2 ± 0.2</td>
<td>54.2 ± 2.6</td>
<td>109.6 ± 1.9</td>
<td>39.8 ± 1.2</td>
</tr>
<tr>
<td>10</td>
<td>0.0 ± 0.0</td>
<td>3.3 ± 0.3</td>
<td>55.4 ± 3.2</td>
<td>108.6 ± 1.0</td>
<td>42.3 ± 1.3</td>
</tr>
<tr>
<td>11</td>
<td>0.0 ± 0.0</td>
<td>1.1 ± 0.2</td>
<td>63.5 ± 9.5</td>
<td>108.7 ± 1.1</td>
<td>41.3 ± 1.7</td>
</tr>
<tr>
<td>12</td>
<td>0.0 ± 0.0</td>
<td>1.1 ± 0.2</td>
<td>54.2 ± 1.9</td>
<td>107.6 ± 2.2</td>
<td>39.5 ± 1.8</td>
</tr>
<tr>
<td>13</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>55.0 ± 1.8</td>
<td>107.4 ± 1.4</td>
<td>39.0 ± 0.8</td>
</tr>
<tr>
<td>14</td>
<td>0.0 ± 0.0</td>
<td>0.2 ± 0.2</td>
<td>55.0 ± 2.0</td>
<td>108.7 ± 2.0</td>
<td>40.5 ± 1.6</td>
</tr>
<tr>
<td>15</td>
<td>0.0 ± 0.0</td>
<td>2.2 ± 0.2</td>
<td>57.3 ± 2.4</td>
<td>108.7 ± 1.9</td>
<td>36.9 ± 1.7</td>
</tr>
<tr>
<td>16</td>
<td>0.0 ± 0.0</td>
<td>4.8 ± 0.2</td>
<td>57.6 ± 3.3</td>
<td>109.1 ± 0.9</td>
<td>46.1 ± 1.3</td>
</tr>
<tr>
<td>17</td>
<td>0.0 ± 0.0</td>
<td>1.0 ± 0.0</td>
<td>55.2 ± 2.9</td>
<td>107.2 ± 1.5</td>
<td>40.7 ± 1.6</td>
</tr>
<tr>
<td>18</td>
<td>0.0 ± 0.0</td>
<td>2.1 ± 0.6</td>
<td>56.2 ± 3.2</td>
<td>108.2 ± 1.5</td>
<td>41.1 ± 1.5</td>
</tr>
<tr>
<td>19</td>
<td>0.0 ± 0.0</td>
<td>3.3 ± 0.6</td>
<td>53.2 ± 1.2</td>
<td>108.3 ± 1.8</td>
<td>41.9 ± 1.6</td>
</tr>
<tr>
<td>20</td>
<td>0.0 ± 0.0</td>
<td>3.0 ± 0.0</td>
<td>57.1 ± 2.4</td>
<td>108.7 ± 1.5</td>
<td>42.0 ± 0.6</td>
</tr>
<tr>
<td>21</td>
<td>0.0 ± 0.0</td>
<td>2.0 ± 0.3</td>
<td>56.2 ± 2.6</td>
<td>110.0 ± 2.0</td>
<td>41.2 ± 1.2</td>
</tr>
<tr>
<td>22</td>
<td>0.0 ± 0.0</td>
<td>1.0 ± 0.0</td>
<td>86.3 ± 29.9</td>
<td>108.8 ± 1.7</td>
<td>46.3 ± 1.4</td>
</tr>
<tr>
<td>23</td>
<td>0.0 ± 0.0</td>
<td>0.3 ± 0.3</td>
<td>54.3 ± 1.7</td>
<td>107.9 ± 1.6</td>
<td>40.8 ± 0.9</td>
</tr>
<tr>
<td>24</td>
<td>0.0 ± 0.0</td>
<td>3.7 ± 0.4</td>
<td>52.2 ± 1.8</td>
<td>108.9 ± 1.3</td>
<td>43.9 ± 0.8</td>
</tr>
<tr>
<td>25</td>
<td>0.0 ± 0.0</td>
<td>4.8 ± 0.4</td>
<td>59.1 ± 5.3</td>
<td>108.7 ± 1.8</td>
<td>38.6 ± 0.7</td>
</tr>
<tr>
<td>26</td>
<td>0.0 ± 0.0</td>
<td>2.2 ± 0.2</td>
<td>82.4 ± 29.4</td>
<td>109.0 ± 1.5</td>
<td>43.5 ± 1.4</td>
</tr>
<tr>
<td>27</td>
<td>0.0 ± 0.0</td>
<td>2.1 ± 0.3</td>
<td>56.3 ± 1.8</td>
<td>107.1 ± 2.5</td>
<td>43.1 ± 0.8</td>
</tr>
<tr>
<td>28</td>
<td>0.0 ± 0.0</td>
<td>3.6 ± 0.3</td>
<td>59.5 ± 5.8</td>
<td>108.3 ± 1.9</td>
<td>42.4 ± 1.3</td>
</tr>
<tr>
<td>29</td>
<td>0.0 ± 0.0</td>
<td>0.7 ± 0.5</td>
<td>55.3 ± 2.8</td>
<td>108.0 ± 1.6</td>
<td>42.3 ± 1.6</td>
</tr>
</tbody>
</table>
Appendix D

Falsification Sampling Supplemental Material

D.1 Experiment 1 - All Good Hypotheses, No Noise

Experiment 1 - All Hypotheses Reasonable, $N = 10$, $T = 25$, No Noise

![Belief vs Hypotheses Graph](image)

**Figure D.1:** Falsification Search increase belief in $H^*$ is 2.3%, with probability > 95% $d=0.47$. 

171
Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 50$, No Noise

**Figure D.2:** Falsification Search increase belief in $H^*$ is 2.5% with probability $> 95\%$ $d=0.72$.

Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 100$, No Noise

**Figure D.3:** Falsification Search increase belief in $H^*$ is 2.2% with probability $> 95\%$ $d=0.69$. 
Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 150$, No Noise

** Figure D.4: ** Falsification Search increase belief in $H^*$ is 2.3% with probability > 95% $d=0.72$.

Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 200$, No Noise

** Figure D.5: ** Falsification Search increase belief in $H^*$ is 2.5% with probability > 95% $d=0.78$.  

173
D.2 Experiment 2 - Mixed Quality Hypotheses, No Noise

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 25$, No Noise

![Graph showing belief levels for $H_0$, $H_1$, $H_2$, and $H_3$ for Control and Proposed methods.]

**Figure D.6:** There is no statistically significant difference in performance.

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 50$, No Noise

![Graph showing belief levels for $H_0$, $H_1$, $H_2$, and $H_3$ for Control and Proposed methods.]

**Figure D.7:** Falsification sampling has greater belief in $H^*$ with average increase in belief is 4.8%, with probability $>95\%$, $d=0.63$. 
Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 100$, No Noise

Figure D.8: Here again there is no statistically significant change in performance.

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 150$, No Noise

Figure D.9: Falsification sampling has greater belief in $H^*$ with average increase in belief is 4.9% with probability $>95\%$, $d=0.67$. 
Figure D.10: Falsification sampling has greater belief in $H^*$ with an average increase in belief is 5.4%, with probability $> 95\%$, $d=0.75$. 

Experiment 2 - Mixed Hypotheses Quality, $N = 10$, $T = 200$, No Noise
D.3 Experiment 3 - All Bad Hypotheses, No Noise

Experiment 3 - All Hypotheses Poor, $N = 10, T = 25$, No Noise

![Bar chart showing belief levels for hypotheses $H_0$, $H_1$, $H_2$, and $H_3$.]

**Figure D.11**: Falsification sampling has greater belief in $H_0$ by 17.8% with 95% probability, $d=1.82$. Falsification sampling has reduced belief in $H_1$ by 9.2% with 95% probability, $d=0.92$. 
Experiment 3 - All Hypotheses Poor, $N = 10, T = 50$, No Noise

**Figure D.12:** Falsification sampling has greater belief in $H_0$ by 16.6% with 95% probability, $d=1.87$. Falsification sampling has reduced belief in $H_1$ by 7.5% with 95% probability, $d=0.94$.

 Experiment 3 - All Hypotheses Poor, $N = 10, T = 100$, No Noise

**Figure D.13:** Falsification sampling has greater belief in $H_0$ by 13.6% with 95% probability, $d=1.37$. Falsification sampling has reduced belief in $H_1$ by 6.2% with 95% probability, $d=0.78$. 

178
Experiment 3 - All Hypotheses Poor, \( N = 10, T = 150, \) No Noise

**Figure D.14:** Falsification sampling has greater belief in \( h_0 \) by 11.0% with 95% probability, \( d=1.15 \). Falsification sampling has reduced belief in \( h_1 \) by 5.2% with 95% probability, \( d=0.81 \).

Experiment 3 - All Hypotheses Poor, \( N = 10, T = 200, \) No Noise

**Figure D.15:** Falsification sampling has greater belief in \( H_0 \) by 10.3% with 95% probability, \( d=1.10 \). Falsification sampling has reduced belief in \( H_1 \) by 4.7% with 95% probability, \( d=0.75 \).
D.4 Experiment 1 - All Good Hypotheses, Noisy Observations

Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 25$, with Noise

![Graph showing belief](image)

**Figure D.16:** Falsification sampling increases belief in $H_1$ by an average of 0.3%, with probability $> 95\%$, $d=0.80$.

Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 50$, with Noise

![Graph showing belief](image)

**Figure D.17:** Falsification sampling and mutual information sampling are statistically indistinguishable at a confidence level of 95%.
**Experiment 1 - All Hypotheses Reasonable, \( N = 10, T = 100 \), with Noise**

Figure D.18: Falsification sampling increases belief in \( H_1 \) by an average of 0.2\%, with probability > 95\%, \( d=0.65 \). There is a reduction of belief in \( H_0 \) with probability > 95\% of 0.1\%, with \( d=0.64 \).

---

**Experiment 1 - All Hypotheses Reasonable, \( N = 10, T = 150 \), with Noise**

Figure D.19: Falsification sampling increases belief in \( H_1 \) by an average of 0.2\%, with probability > 95\%, \( d=0.63 \).
Experiment 1 - All Hypotheses Reasonable, $N = 10, T = 200$, with Noise

**Figure D.20:** Falsification sampling increases belief in $H_1$ by an average of 0.1%, with probability $> 95\%$, $d=0.68$. 

![Belief vs Hypothesis with Control and Proposed](image)
D.5 Experiment 2 - Mixed Quality Hypotheses, Noisy Observations

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 25$, with Noise

![Bar chart showing beliefs for hypotheses $H_0$, $H_1$, $H_2$, $H_3$]

**Figure D.21:** Falsification sampling and mutual information sampling are statistically indistinguishable at a confidence level of 95%. Both algorithms select the best hypothesis.
Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 50$, with Noise

![Bar Chart](chart1)

**Figure D.22:** Falsification sampling and mutual information sampling are statistically indistinguishable at a confidence level of 95%. Both algorithms select the best hypothesis.

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 100$, with Noise

![Bar Chart](chart2)

**Figure D.23:** Falsification sampling and mutual information sampling are statistically indistinguishable at a confidence level of 95%. Both algorithms select the best hypothesis.
**Figure D.24:** Falsification sampling and mutual information sampling are statistically indistinguishable at a confidence level of 95%. Both algorithms select the best hypothesis.

**Figure D.25:** Falsification sampling has greater belief in the best hypothesis with an average increase of 0.2% with probability $>95\%$ and $d=0.63$. 

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 150$, with Noise

Experiment 2 - Mixed Hypotheses Quality, $N = 10, T = 200$, with Noise
Experiment 3 - All Bad Hypotheses, Noisy Observations

Experiment 3 - All Hypotheses Poor, $N = 10$, $T = 25$, with Noise

Figure D.26: Falsification sampling and mutual information sampling have statistically indistinguishable belief in $H^*$ at 95% confidence level. However, both would correctly identify $H_0$ as the correct hypothesis.
Experiment 3 - All Hypotheses Poor, $N = 10, T = 50$, with Noise

Figure D.27: Falsification sampling is more likely to believe in $H_0$ with an increase of 1.7% with probability $> 95\%$, $d=1.14$. Falsification is less likely to belief in $H_7$ with a decrease of 1.0%, with probability $> 95\%$, $d=-1.39$.

Experiment 3 - All Hypotheses Poor, $N = 10, T = 100$, with Noise

Figure D.28: Falsification sampling is more likely to believe in $H_0$ with an increase of 1.0% with probability $> 95\%$, $d=1.03$. 
**Figure D.29:** Falsification sampling is more likely to believe in $H_0$ with an increase of 0.6% with probability $> 95\%$, $d=1.09$. Falsification is less likely to believe in $H_7$ with a decrease of 0.3%, with probability $> 95\%$, $d=-1.04$. 

**Figure D.30:** Falsification sampling is more likely to believe in $H_0$ with an increase of 0.7% with probability $> 95\%$, $d=1.91$. 

Experiment 3 - All Hypotheses Poor, $N = 10, T = 200$, with Noise
Bibliography

Bruce Abramson. The expected-outcome model of two-player games. 1987. 4.2


Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In COLT, pages 39–1, 2012. 5.1

Pablo F Alcantarilla, Simon Stent, German Ros, Roberto Arroyo, and Riccardo Gherardi. Street-view change detection with deconvolutional networks. In Robotics: Science and Systems, 2016. 4.2


John E Bares and David S Wettergreen. Dante ii: Technical description, results, and lessons


Dean Bergman, Brian J Glass, Thomas Stucky, Kris Zacny, Gale Paulsen, and Chris McKay. Obtaining vibration data for autonomous health monitoring of interplanetary drills. In *AIAA SPACE 2016*, page 5485. 2016. 5.4


Steve Chien, Rob Sherwood, Daniel Tran, Rebecca Castano, Benjamin Cichy, Ashley Davies, Gregg Rabideau, Nghia Tang, Michael Burl, Dan Mandl, et al. Autonomous science on the eo-1 mission. 2003. [2.2.1]


Tara A Estlin, Benjamin J Bornstein, Daniel M Gaines, Robert C Anderson, David R Thompson, Michael Burl, Rebecca Castaño, and Michele Judd. Aegis automated science targeting for the


Daniel M Gaines, Tara Estlin, Steve Schaffer, Rebecca Castano, and Alberto Elfes. Robust and opportunistic autonomous science for a potential titan aerobot. 2010.


Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis

Ross D King, Jem Rowland, Wayne Aubrey, Maria Liakata, Magdalena Markham, Larisa N Soldatova, Ken E Whelan, Amanda Clare, Mike Young, Andrew Sparkes, et al. The robot scientist adam. *Computer*, 42(8), 2009a. 2.4

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009b. 2.4


Pat Langley, Herbert A Simon, Gary L Bradshaw, and Jan M Zytkow. Scientific discovery: computational explorations of the creative process. 1987. 2.4


Leonid A Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. 2.4

James Lind. *A Treatise on the Scurvy: In Three Parts, Containing an Inquiry Into the Nature*,
Causes, and Cure, of that Disease. A. Millar, 1757.


Charles Sanders Peirce and Joseph Jastrow. On small differences in sensation. 1884.


RZ Sagdeev and VI Moroz. Project vega first stage-missions to venus. Pisma v Astronomicheskii Zhurnal, 12:5–9, 1986.


Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. Biometrika, pages 1–85, 1918.


MA Wiering and Jürgen Schmidhuber. Solving pomdps with levin search and eira. 1996. 2.4


