ENTRAINING A ROBOT TO ITS ENVIRONMENT WITH AN ARTIFICIAL CIRCADIAN SYSTEM

A Dissertation
Presented to
The Academic Faculty

by

Matthew J O'Brien

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineer

Georgia Institute of Technology
August 2021

ENTRAINING A ROBOT TO ITS ENVIRONMENT WITH AN ARTIFICIAL CIRCADIAN SYSTEM

Approved by:

Dr. Ronald C. Arkin, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ayanna Howard
College of Engineering
Ohio State University

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Céderic Pradalier
School of Interactive Computing
Georgia Institute of Technology

Date Approved: May 04, 2021

To my parents and fiancé, who supported and encouraged me through this long journey

ACKNOWLEDGEMENTS

The person whom I'd like to thank the most is my advisor, Dr. Ronald C. Arkin, for providing the incredible opportunity to work on both this dissertation and other research projects during my time at Georgia Tech. His thoughts helped inspire the research questions driving this work, and his advice on technical work, scientific research, and writing have helped make this dissertation far better.

I also wish to thank my committee: Dr. Ayanna Howard, Dr. Cédric Pradalier, Dr. Magnus Egerstedt, and Dr. Zsolt Kira. Despite their busy schedules, each promptly met with me when asked, providing critical feedback to this work. Specific thanks to Dr. Pradalier for being so flexible on scheduling when committee members were spread across several time zones.

Two members of the Mobile Robot Lab also deserve special thanks: Michael Pettinati and Shu Jiang. Both of you helped teach me how to use lab software and hardware when I joined, discussed technical issues ranging from code bugs to experimental designs, and joined me for numerous enjoyable lunches. I consider you not just colleagues, but good friends. Speaking of friends, I'd like to mention Alex Chang who I met during my first visit to Georgia Tech, and ended up being my roommate for around five years. As we went through our entire PhD career, you helped make it more fun.

I must thank my parents, who have been supportive and encouraging for quite a long time. Without the time they dedicated to driving me to clubs, camps, and colleges when I was younger, and without the encouragement they provided to push myself, I would

likely not have pursued a PhD. They are always expressing how proud they are of me, and I'm incredibly appreciative of all they have done for me.

Lastly I would like to thank my fiancé, who has been my partner throughout my PhD. From making a quals survival kit to reviewing chapters of this dissertation, she has provided both love and help whenever needed. Her work ethic and dedication to her own goals were also an inspiration, something I looked to for motivation when it was hard to find.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

SUMMARY xiii

CHAPTER 1. Introduction 1
1.1 Research Questions 3
1.2 Contributions 5
1.3 Outline 5

CHAPTER 2. Related Work 7
2.1 Circadian Rhythms 7
2.2 Time Series Forecasting 10
2.3 Long-term Autonomy 15
2.4 Markov Decisions Processes and Reinforcement Learning 16
2.5 Model Predictive Control 19

CHAPTER 3. The Artificial Circadian System 22
3.1 Dynamic Environments 22
3.2 The Entrainment Problem 24
3.3 Time series Forecasting 25
3.3.1 Approach 25
3.3.2 Forecasting Example Application 31
3.4 Action Selection 35
3.4.1 The Circadian Function 35
3.4.2 Circadian Function Example Application 39
3.4.3 Behavior Activation 41
3.4.4 Behavior Activation Example Application 44
3.5 Summary 45

CHAPTER 4. A notional agricultural experiment 47
4.1 Construction of the Agricultural Testbed 48
4.1.1 Simulated version 50
4.2 Experiment Design 51
4.2.1 Environmental Variables 51
4.2.2 Charge Behavior 55
4.2.3 Pest Behavior 57
4.2.4 Weed Behavior 58
4.2.5 Scenarios and Measures 59
4.2.6 Procedure Details 62
4.3 Results 63

4.3.1 Scenario 1: Local Sensing 63
4.3.2 Scenario 2: Slow actuation 65
4.3.3 Scenario 3: Dynamic weather 67
4.4 The Scope of the ACS 69
4.5 Conclusions 72

CHAPTER 5. Robustness to changing dynamics 74
5.1 Robustness in the Artificial Circadian System 74
5.1.1 Identifying Modelling Error 75
5.1.2 Adjusting Action Selection 80
5.1.3 Recovery 85
5.2 Robust Experiment Design 94
5.2.1 Environmental Variables 95
5.2.2 Behaviors 98
5.2.3 Scenarios, Measures, and Hypotheses 100
5.2.4 Procedure Details 103
5.3 Results 104
5.3.1 Scenario 1: Shifted Disruption 104
5.3.2 Scenario 2: Random Disruption 107
5.3.3 Scenario 3: Permanent Change 109
5.4 Conclusions 111

CHAPTER 6. Conclusion 114
6.1 Research Questions Revisited 114
6.2 Contributions 117
6.3 Future work 120
6.4 Final words 121

APPENDIX A. Forecasting Model Parameters 122
A.1 Chapter 4 Pest Forecasting Model 122
A.2 Chapter 5 Pest Forecasting Model 123

References 125

LIST OF TABLES

Table 3.1	Algorithm to apply discrete time series models in real time on autonomous agents.	29
Table 4.1	Charge Behavior Activation Components.	55
Table 4.2	Pest Behavior Activation Components.	57
Table 4.3	Weed Behavior Activation Components	59
Table 4.4	Summary of experiment hypotheses for each scenario.	61
Table 5.1	Algorithm to calculate the scaled error of a forecast.	79
Table 5.2	Forecast Usage in ACS Modes.	90
Table 5.3	Algorithm to monitor forecast accuracy and implement robust methods.	93
Table 5.4	Charge Behavior Activation Components	98
Table 5.5	Pest Behavior Activation Components.	99
Table 5.6	Shifted Disrupted Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Robust ACS condition saw a statistically significant decrease in the pest population compared to the No ACS (reactive) and Basic ACS conditions.	104
Table 5.7	Random Disruption Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Basic-ACS and Robust-ACS conditions both had significantly lower pest population from the No ACS condition, and were not significantly different from each other.	107
Table 5.8	Permanent Change Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Robust ACS condition had a pest population significantly lower compared to the No ACS (reactive) and Basic ACS conditions.	109
Table A.1	Smoothing parameters and initial states for Chapter 4 pest model.	122
Table A.2	Parameters for Chapter 5 pest model.	124

LIST OF FIGURES

Figure 2.1	Basic circadian system diagram based on	8
Figure 2.2	Historical data of pedestrian traffic (black) and the forecast of this traffic (blue) 24 hours into the future using the TBATS model, an exponential smoothing based method (De Livera, Hyndman, & Snyder, 2011).	12
Figure 2.3	MPC controller leveraging forecasting from.	21
Figure 3.1	The representation of the forecastable state over time, broken into the past history ($\boldsymbol{S_P}$), present measured ($\boldsymbol{S_M}$), and future forecasted ($\boldsymbol{S_F}$).	28
Figure 3.2	Traffic data used for this example.	32
Figure 3.3	Traffic data (black) plus two days of forecasted traffic (blue).	35
Figure 3.4	Visualization of the predicted performance function, P, which acts as a sliding window over the forecasted state, determining the performance for executing a behavior at a certain time t.	37
Figure 3.5	The parameter n adjusts the bias of the circadian function towards the maximum performance, with higher values of n making a sub-maximal performance provide less activation.	39
Figure 3.6	Forecasted state, predicted performance, and circadian activation for one environmental cycle into the future. The scale for predicted performance P is not shown, but is aligned against the circadian function for easier comparison.	41
Figure 3.7	The total activation of the delivery behavior when using the circadian function (black) and when not using the circadian function (purple). Generated by applying equation 3.14, with *time_since_last* starting at 24 hours, and the real traffic S for a time step equal to the forecasted traffic. The green shaded regions mark areas where the circadian function significantly changes the total behavior activation.	45
Figure 4.1	The Neato Botvac platform used in the physical experiment, with the Raspberry Pi controller and camera mounted. The charging station is shown behind it.	48

Figure 4.2	The physical testbed. Each row shown has artificial plants on both sides. The green shrubs shown are decorative fakes, no plants were harmed in this work.	50
Figure 4.3	The gazebo simulation of the physical testbed. In this image, the locations of individual artificial plants (P1 through P16) and quadrants that share pest populations (Q1 through Q4) are labeled.	51
Figure 4.4	Solar irradiation data for several consecutive days. There are examples of clear and cloudy weather.	52
Figure 4.5	Solar irradiation data for a full year. The max follows a smooth curve based on the seasonal change, while weather causes regular but random decreases in available energy.	52
Figure 4.6	Pest activity for one quadrant over seventeen days. Decreases in activity were caused by the agent removing pests.	53
Figure 4.7	Pest and weed levels for scenario 1. The colored bars are the simulated results, with standard error shown. The horizontal lines show the results of the hardware trials.	63
Figure 4.8	Histograms of the pest behavior activation times for each quadrant.	64
Figure 4.9	Pest and weed levels for scenario 2. The colored bars are the simulated results, with standard error shown. The horizontal lines show the results of the hardware trial.	66
Figure 4.10	Histograms of the pest behavior activation for each quadrant.	67
Figure 4.11	Histograms of the charge behavior activation.	68
Figure 4.12	An agent will not need, or be able, to entrain behavior to environment dynamics that are much faster or slower than its actions. For dynamics at a similar time-scale, the timing of actions may have a significant impact.	70
Figure 5.1	(a) shows and example time series with a disruption between hours 125 and 140. (b) shows the forecast error. The blue and red forecasts are used to calculate the SE at the time steps marked by the vertical blue/red lines.	79
Figure 5.2	The error from the example in Figure 5.1 with circadian weight Wc plotted.	84
Figure 5.3	The function Wc plotted for several additional values of the midpoint (m) and growth rate (k) parameters. Lower midpoint (m)	85

and gain (k) values means it is more likely for the circadian weight Wc to be reduced without turning off entirely.

Figure 5.4 A hypothetical disruption event, where the dynamics on day three return to the regular pattern seen before the disruption in day two. The forecast from the current time is influenced by the disruption, and may have more error then a forecast generated with data before the disruption. 86

Figure 5.5 State machine for the robust ACS. The robust mode generates and tests the outlier forecast to see if it improves accuracy. The outlier mode provides this outlier forecast to the action selection system. The change mode, if reached, will notify supervisors that a significant change in the environment has happened, and intervention is needed. 89

Figure 5.6 The pest activity of one quadrant over several days is shown above in black. The forecast forward from the current time is in blue, while the forecast from 24 hours ago is shown in red. The forecast model is able to do a good job predicting pest activity, even including pest removals by the agent which create the dip in pest activity just after day 6. 98

Figure 5.7 Example pest history (black) from the shifted disruption scenario. The ‘start’ and ‘end’ lines mark the period the dynamics are changed. The peak of activity around day 9 is wider and offset from the regular cycle. Though the dynamics returned to normal, the forecasts (shown in red/blue) are still degraded. 101

Figure 5.8 Example pest history (black) from the random disruption scenario. The ‘start’ and ‘end’ lines mark the period the dynamics are changed. Pest activity during this disruption is randomly shuffled hour-by-hour. The accuracy of forecasts (red/blue) generated afterwards are again degraded. 101

Figure 5.9 Example pest history (black) from the permanent change scenario, where the cycle’s duration is reduced. The ‘start’ line marks where the dynamics change. The past forecast (red) was generated with data just after the change to the environment and highlights how the new pest activity is offset from the predicted activity. 102

Figure 5.10 Pest removals per day for one trial. The vertical lines mark when the environment was shifted. In this trial, the basic ACS showed a sharp decline in daily removals during the disruption, due to its forecast being out-of-sync with the shifted environment. This did not happen for the no-ACS or robust-ACS conditions. 105

Figure 5.11 The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the shifted dynamics. In this trial/quadrant, the outlier was quickly identified (see the blue region beginning in day 6, identifying that the robust mode started). An outlier-forecast successfully generated accurate predictions and transitioned the robust ACS to outlier mode (shown as the yellow region) until the normal forecast error returned to adequate levels. 106

Figure 5.12 The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the shifted dynamics. In this trial/quadrant, the outlier was identified late, and a good outlier-forecast was never generated. However, the robust-ACS still reduced the weight of the circadian component after the disruption. Through most of day 7 to day 10, Wc is significantly reduced due to the increased forecast error. This mitigates the worst outcomes from inaccurate forecasts. 107

Figure 5.13 The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the random dynamics. Due to the naïve forecast error increasing at the same time, the SE does not increase during the random disruption. Instead, SE spikes after the dynamics return to normal, but when data from the disruption is still impacting forecasting. 109

Figure 5.14 The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The black vertical line indicates when the dynamics change. In this trial, the robust-ACS quickly enters robust mode after the dynamics change and error increases. Once enough time has passed, it transitions to change mode, representing that it has detected a long-term change to the environment. 110

Figure 5.15 The total pest population averaged over all trials. Before the environment change, marked by the vertical red line, both the basic and robust-ACS conditions diverge from the no-ACS condition. After the change, the robust-ACS looks to be approaching the no-ACS condition, while the basic-ACS sharply rises above both. 111

SUMMARY

As robots become persistent agents in a complex and dynamic world, they must deal with changing environments. This challenge has grown research in long-term autonomy, with particular focus on localization and mapping in dynamic environments. Less attention has been paid to learning these dynamics to adapt or entrain an agent's behavior to them.

Inspired by circadian systems in nature, this dissertation seeks to answer how a robotic agent can both learn the regular cycles and patterns that exist in many environments, and how it can exploit that knowledge. In this work, relevant environmental states are modeled as time series and forecasted into the future. These forecasts are used to estimate the utility of executing some behavior at any point in time during the dominant environmental cycle. The relative utility of executing a behavior at the current time, compared to the potential utility for waiting until later, is passed as one component in an activation-based action selection system. While other components still impact what behaviors execute when, the 'circadian' component derived from forecasts biases the behavior to execute at the better times in the environmental cycle. This approach was dubbed the artificial circadian system.

As forecasting the future is inherently unreliable, methods to make the approach robust to degrading forecast accuracy are presented. A unitless error measure is used to adapt the weight of the forecasting component in action selection, allowing an autonomous agent to leverage forecasts when they are good, and fall back onto reactive strategies when forecasts fail. As time series models rely on the history for predictions, a temporary disruption to the environment can potentially degrade forecast accuracy for many cycles.

Methods to create modified forecasts which exclude potential outlier data are presented, and these forecasts are leveraged only if they improve accuracy.

The ideas in this work were validated using an experimental test bed designed to approximate a precision agricultural task, where a solar powered robot monitored individual plants for pests and weeds. The artificial circadian system was shown to effectively entrain the agent's behavior to the dynamics of its environment in some cases, improving performance. It was also noted that dynamics not on the same time-scale as the robot's actions could not be exploited, even with forecasted knowledge. The artificial circadian system was reliably able to detect deviations in the environment and remove the influence of forecasts when their accuracy degraded. Successfully removing outlier data to generate better forecasts was less consistent, but achieved in a significant portion of trials.

CHAPTER 1. INTRODUCTION

"Change is the law of life and those who look only to the past or present are certain to miss the future." – John F. Kennedy

The natural world is in constant change, and within those changes exist cycles that occur at a variety of time scales. The most significant of these is the 24-hour solar cycle. It is not surprising that life has evolved adaptations for this regular, ever-present change. Circadian clocks are biological oscillators that synchronize behavioral and metabolic processes to the solar cycle. They appear in all types of life; from complex mammals like humans (Czeisler & Gooley, 2007), to plants (Gardner, Hubbard, Hotta, Dodd, & Webb, 2006), to single cell bacteria (Johnson & Golden, 1999). Circadian systems entrain behavior to the dynamics of the environment, allowing animals to take the right actions at the right times, before their needs (internal state) or senses (perceptual state) could inform their behavior. "Circadian rhythmicity of behavior represents an animal's information, or one is tempted to say, knowledge about a particular feature of its environment… and what to do about it." (Oatley, 1974)

Adaption to a changing environment is not exclusive to circadian processes. Human activity expresses a strong daily cycle, and people use explicit models of this activity to guide behavior. Someone may head out early to beat the lunch rush, or wait to leave until traffic has died down. This extends to organizations, businesses, and even governments who plan and act based on the predicted outcomes of reliable cycles in the environment and human activity. With a trait so prevalent in both the natural world and human society,

it is prudent to explore how and when such a concept should be implemented into a robotic agent.

Until recently, the only persistent robots used were operated in highly engineered, static environments. However, in 2016 the non-industrial robotic market surpassed the industrial robotic market for the first time (Tractica Research, 2017). Now, more than ever, robots are leaving the clean and controlled environments of the factory floor and entering a complex dynamic world. Many of these robots are persistent, designed to continuously operate. Robotic vacuum cleaners for inside homes and agricultural robots for both commercial and personal use exist and are sold (Franklin Robotics, 2020). In the near future, autonomous cars and delivery robots may begin to deal with the same daily traffic patterns that humans do. Persistent robots in a variety of domains will face dynamics in their environment over multiple time scales, generated both from the natural world and human behavior.

Of special interest are Slowbots, or slow robots (Arkin & Egerstedt, 2015), which offer unique advantages in reducing energy usage and wear, especially for persistent tasks. There are, of course, downsides to being slow. Such as a reduced ability to rapidly react to changes in the environment. The research reported here is especially well-suited for Slowbots. Relative to faster agents, a Slowbot will perceive environmental change as faster, making it more difficult to manage or respond to this change. If a Slowbot can predict future changes in its environment, it can take action before critical changes occur, bypassing some of the limitations of being slow. In contrast, a robotic agent that can respond rapidly may be able to adequately adapt to a changing environment through pure

reactive control. This work focuses explicitly on the scenarios where reactive response is not adequate, and Chapter 4 will explore this scope in more detail.

Inspired by circadian systems in nature and motivated by the growing number of persistent robotic platforms, this thesis investigates how an autonomous robotic agents, particularly slow and persistent ones, can learn and adapt to the dynamics of the environments they inhabit. In other words, how an autonomous agent can entrain itself to its environment. This work will show how such an approach can help an agent compensate for limitations in actuation and sensing. Over time, the dynamics of the environment itself may change due to either short-term perturbations (e.g. extreme weather, special events) or permanent changes (e.g. a new pest species invading, new roads being built, etc). Methods for responding to changes in the dynamics will also be discussed. This adds an element of robustness as the agent explicitly checks the validity of its forecasts over time.

1.1 Research Questions

To provide a clear structure for investigating this topic, this section introduces a set of research questions. The overall aim of the proposed work can be summarized in the principal research question:

> *How can the long-term dynamics of an environment be modeled, predicted, and exploited by a slow and persistent autonomous robot?*

This can be broken down into relatively independent tasks for investigation, described by four subsidiary research questions:

1. How can the long-term dynamics of an agent's environment be characterized or modeled?

Given a relevant time-varying state variable, what equations or algorithms could be used to model its dynamics? How is the model generated? How can predictions or forecasts of its future values be generated? When should the model change due to new information?

2. How can individual behaviors, and the set of active behaviors, adapt to changes in the environment using the knowledge of environmental dynamics?

Given a model of the environment, and predictions of that environment's future state, what should a robot do? How and when can individual behaviors adjust based on future forecasts? How and when can the choice of active behaviors, future task schedule, or other action-selection processes change based on the forecasted state?

3. Can such a system be made robust to changes in the dynamics, due to either short-term perturbations or long-term environmental change?

The environment dynamics themselves will change over time. How can the agent identify when this has occurred? What actions should the agent take when its environmental models are not accurate? How can the agent reset or re-entrain its model to the new dynamics if the environment has changed, or back to the old dynamics after random perturbations?

4. In what contexts is entraining to a dynamic environment useful for a robotic agent?

Both reactive behavior and hard-coded schedules have been used successfully to deal with changing environments. When are these approaches not enough, and why do they fail? How does the proposed approach address their shortcomings?

1.2 Contributions

In answering the above research questions, the following contributions to the area of mobile autonomous robotics are expected:

- An architecture for an artificial circadian system (ACS) that enables an autonomous agent to model, forecast, and adapt to dynamic environments. This includes:
    - An approach to applying time series forecasting in a real-time context
    - Incorporation of forecasts into action-selection mechanisms
- Techniques for robustness – allowing an agent to detect and respond when the environment deviates from the expected dynamics.
- Analysis of when such an approach is useful, and when simpler alternatives (such as reactive behavior) will be enough.

1.3 Outline

The remainder of this dissertation is as follows: Chapter 2 discusses several fields of research that this work either pulls from or relates to. Chapter 3 details the analysis of the problem and design of the artificial circadian system, addressing the first two subsidiary research questions. In Chapter 4 the agricultural test bed will be presented along with experimental results. In the context of these results, the fourth subsidiary research question will be addressed. An approach to robustly dealing with changing dynamics will be covered

in Chapter 5, along with experimental validation of the approach on a mock-up agricultural test. This addresses the third subsidiary research question. The sixth and final chapter of this document is the conclusion, where the work will be summarized and concluding remarks added.

CHAPTER 2. RELATED WORK

This thesis draws on ideas from several fields. Within research on long-term autonomy for robotics, similar work can be found but is relatively scarce. Thus, the majority of this chapter is on various topics that inspired, were leveraged, or considered by this work. First, circadian systems in nature will be overviewed before moving on to discuss the modeling framework used, statistical time series forecasting. Extra detail will be spent on covering time series forecasting as it's not a standard approach within the field of robotics research. Then approaches in long-term autonomy, both within robotics and outside of it, will be covered. Next approaches in reinforcement learning for dealing with time varying environments will be presented. Finally, model predictive control and how it relates to this work will be overviewed.

2.1 Circadian Rhythms

In many organisms, there exists a special set of biological rhythms that are both endogenous (internally generated) and entrained to the environment. While circadian rhythms are the most well studied and perhaps impactful example, there are three other significant "circa-rhythms" that have been identified in nature. Each corresponds to a unique temporal niche, where each niche is created by a different cyclic geophysical process. They are the circa-tidal, circa-lunar, and circannual rhythms (Aschoff J. , 1981). As circadian rhythms are the most well-known they will be the focus moving forward, but these principles are not specific to interacting with the solar cycle.

A circadian clock has three primary components, shown below in Figure 2.1. First is the internal oscillator which tracks the passage of time and approximately aligns with the period of an environmental cycle. This internal oscillator is created by genes whose protein products regulate said gene's production in a feedback loop (Hardin, Hall, & Rosbash, 1990). Second is the sensory input channel which allows the oscillator to entrain to the environment (through impacting the protein/RNA production cycle). Entrainment is a key property of circadian systems. A "free-running" circadian oscillator can have a notable difference in period from the environmental cycle it is tracking. Similar to a mechanical oscillator, entrainment drives the circadian oscillator at the relevant environmental frequency. It also ensures the circadian system will resync even after extreme disturbances (Roenneberg, Daan, & Merrow, 2003). Finally, there is the output channel, where the current state of the circadian rhythm drives both metabolic processes and behavior.

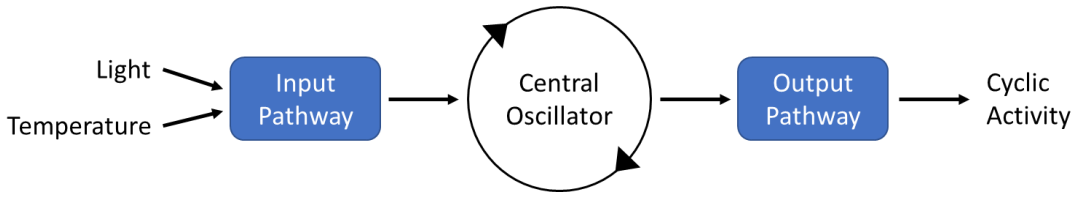

Figure 2.1: Basic circadian system diagram based on (Gardner, Hubbard, Hotta, Dodd, & Webb, 2006)

Circadian systems have been found to provide benefits in many different contexts. A study on fledglings showed that those who left the nest at a species-specific time fell victim to predation less often. By synchronizing with other juveniles, survival was increased (Tinbergen & Daan, 1980). In some parasitoids different related species have peaks of activity at different times of the day, creating a temporal partitioning helps reduce competition (Fleury, Allemand, Vavre, Fouillet, & Bouletreau, 2000). Circadian clocks are used to track the changing length of day to initiate migration in birds, and hibernation in mammals (DeCoursey, 2004). Experiments suggest that the evolutionary benefits of a

circadian rhythm exist primarily in natural dynamic environments. Under controlled laboratory conditions, a species of ground squirrels with their circadian clocks surgically removed had no increase in mortality. When released to a semi-natural enclosure with feral cats, however, they were preyed upon at a significantly higher rate compared to control groups (Paranjpe & Sharma, 2005). There are also contexts where animals with circadian clocks exhibit constant or ultradian (cycle with period shorter than a day, but longer than an hour) activity in environments that are constant, such as animals in polar environments and migratory birds (Bloch, Barnes, Gerkema, & Helm, 2013).

Note that it is possible for cyclic behavior to be generated as a direct, or reactive, response to a cyclically changing environment. Yet, circadian rhythms are not merely responses to periodically changing inputs from the environment, but come from internal processes of the organisms (Aschoff J. , 1981). A circadian system can be considered a model of the dynamics of the organism's environment, representing implicit knowledge about how it is expected to change. An agent or organism can gain value from having even simple models of complex processes (tides, seasons, etc.) in the environment. The literature on circadian systems in nature supports the idea that as robotic agents move out of controlled environments and into dynamic worlds, they too will be able to benefit from entraining activity to their relevant environmental cycles.

Circadian clocks, however, may not be an effective model for how a robotic agent should entrain to an environment. As a circadian clock is a cyclic or phase-only model, it faces two limitations. First is the inability to represent non-cyclic dynamics. For example, traffic has a strong daily pattern, but also has short term surges or trends that can be modeled to some degree. E.g., a traffic jam may be expected to last at least a certain number of hours. Second is the inability to represent the amplitude of the relevant state. In some contexts, the relative values of the state at different times may be critical. Our current understanding of circadian clocks also provides no guidance on how an agent should act

given some model of the environment. The impact of output pathways have been tuned over millennia by evolution, and do not provide a systematic method to determine the correct actions. While circadian rhythms provide support that the work in this dissertation will be useful for persistent agents, a more structured method to model the environmental dynamics, and act based on its forecasted states, must be developed.

2.2 Time Series Forecasting

Time series modeling and forecasting covers the tools and techniques of forecasting (or predicting) future values of some variable based on past values of that variable. To make forecasting possible, it is assumed time series data has a "spatial" property (observations closer in time are more related) and a "temporal" property (past values can impact future values, not vice versa). Time series models are built on these assumptions. Approaches to time series forecasting can be broadly divided into statistical methods and machine learning methods.

The classical approach to time series modeling uses statistical methods. These methods create models that explicitly represent the relationship between past data and future values. Statistical approaches can be divided into two categories. The first is exponential smoothing, which contains the oldest and most widely applied approaches to forecasting due to both their simplicity and accuracy in real applications (Gardner E. S., 1985). The core idea in exponential smoothing is to forecast the next value in a time series as the weighted average of previous values where, specifically, the weight of a previous value decreases exponentially with time. The basic form, referred to as simple exponential smoothing is: (notation used from (Hyndman R. , Koehler, Ord, & Snyder, 2008))

$$\hat{y}_{t+1} = l_t \tag{1.a}$$

$$l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{1.b}$$

Here y_t is the time series. The subscript designates the time step. The level, l_t, updates each time step based on the weighted average of the previous level, and the measured value of the time series. The parameter α specifies the weight of each. The impact of a previous measurement, y_{t-k}, on the current level is thus weighted by $\alpha(1-\alpha)^k$. Finally, the forecasted time series $\hat{y}_{t+1}$ is assumed to be the current level. This method can be extended past a simple estimate of the current level to include knowledge of a trend component (slowly changing average) and seasonal components (cyclic change with known period):

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+h^+} \quad (2.a)$$
$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \quad (2.b)$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \quad (2.c)$$
$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \quad (2.d)$$

Here, b_t is a trend component, modelling a steady change over time as the average difference between the level of a time series at two time-steps. Note that the trend component also exponentially weights past measures of trend to provide a current best estimate. The trend component influences the forecast of future values as well as the estimation for the current level, which is the previous level plus the trend. The seasonal component is s_t, a vector of length m (the period) with an estimated seasonal value for every time-step. The factor h^+ ensures that the seasonal index comes from the last sampled period, not a forecasted period, if more than one period is being forecasted. The parameters γ and β are weights. Exponential smoothing has been extended in various other ways, with other methods for modelling trend and seasonal components, incorporating regressor or explanatory variables, etc. Figure 2.2 shows an example forecast for real pedestrian traffic data with a strong seasonal component. Perhaps the most important extension is its incorporation into a state-space model to provide a true statistical framework (Gardner E. S., 1985). This allows for the generation of forecast intervals along with point forecasts and allows the use of model selection criteria. For a detailed review of exponential smoothing in a state-space framework, see (Hyndman R. , Koehler, Ord, & Snyder, 2008).

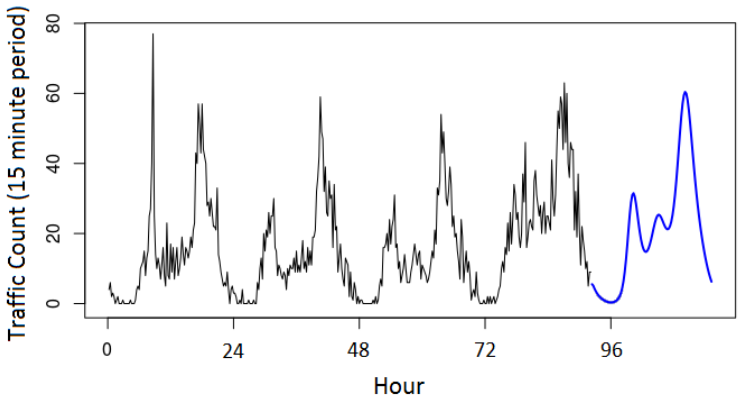

Figure 2.2: Historical data of pedestrian traffic (black) and the forecast of this traffic (blue) 24 hours into the future using the TBATS model, an exponential smoothing based method (De Livera, Hyndman, & Snyder, 2011).

The second major branch for time series forecasting is based around the ARMA model and the Box-Jenkins method (Box, Jenkins, & Reinsel, 2008). ARMA (autoregressive moving average) models focus on the autocorrelations in the data, and are able to approximate a large class of stationary time series. An ARMA model is composed of two main components. First, an auto-regressive (AR) model, where the relevant variable depends on its past values, and an error term, AR(p):

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-1} + e_t \tag{3}$$

Where again, y_t is the time series variable, and ϕ_p are parameters which weight the impact of each previous time step. Second, a moving-average (MA) model, where the relevant variable is the average of the past error (or noise) terms, MA(q)

$$y_t = e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} \tag{4}$$

Here, θ_q are the weights of previous error terms, e_t. These can be combined to get the full ARMA(p,q) model…

$$y_t - \phi_1 y_{t-1} - \cdots - \phi_p y_{t-1} = e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} \tag{5}$$

As this is a model for stationary time series, time-varying components like trend and seasonality must be removed before the model is applied. The technique of differencing is commonly applied. Differencing is the approach of subtracting the value at each time step from a previous time step, to generate a new differenced time series without trend. For seasonality, the subtracted value is not one time-step in the past, but instead an entire period of the season in the past. Backshift notation provides a convenient way to handle differencing. By defining the operator B to shift a variable one timestep into the past (equation 6.a), we can used basic algebraic manipulations to represent both repeated differencing (equations 6.c and 6.d) or seasonal differences (equation 6.e).

$$By_t = y_{t-1} \tag{6.a}$$
$$y_t - y_{t-1} = y_t - By_t = (1 - B)y_t \tag{6.b}$$
$$B^2 y_t = y_{t-2} \tag{6.c}$$
$$(y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t \tag{6.d}$$
$$(1 - B^m)y_t = y_t - y_{t-m} \tag{6.e}$$

As with exponential smoothing, there are many extensions to this basic ARMA model (state space forms, multivariate modeling, etc) which go beyond the scope of this summary. For a thorough introduction to ARMA based methods, see (Brockwell & Davis, 2006). An interesting note is that a large class of ARMA models and exponential smoothing models are equivalent (McKenzie, 1984).

A critical part of getting good forecasts is model selection and parameter estimation. The latter is much simpler in practice. Parameters for both exponential smoothing and ARMA based approaches can generally be found using maximum likelihood estimation. Good libraries are available that provide automated methods to do so for a variety of the standard techniques (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models, 2019). Model selection is not as straight forward. Information criterion can be used to guide the choice, but are not always adequate. In general, analysis of the data being modeled can reveal the

relevant dynamics. Then an adequate model, one that can represent those dynamics and ideally nothing more, can be selected.

An alternative to the statistical approaches described above is to apply more general machine learning (ML) methods for the forecasting problem. Artificial neural networks (ANNs) have received particular focus in the literature. The problem of forecasting a time series is readily applied to an ANN, and as general non-linear function approximators, ANNs have the potential to model very complex processes without any understanding of the system needed by the designer (Zhang, Patuwo, & Hu, 1998). Other machine learning approaches, such as deep learning (Qiu, Zhang, Ren, Suganthan, & Amaratunga, 2014), support vector machines (SVMs), and genetic programming (GP) (Wang, Chau, Cheng, & Qiu, 2009) have seen success when used for forecasting. However, the accuracy of these ML methods has not surpassed statistical ones in independent testing, despite greater computational requirements (Makridakis, Spiliotis, & Assimakopoulos, 2018). Often the fit of the model to the training data was better with the ML methods, which points to worse performance being a case of overfitting. Interestingly, this is a pattern that has also been found within statistical methods, with simple approaches often outperforming more sophisticated models (Makridakis & Hibon, 1979). More research is required to see if machine learning can better address the unique challenges in time series forecasting.

Time series forecasting will be the basis of how robotic agents in this work understand the environment and how it changes. Natural environments often have features driven by difficult to measure variables, such as the availability of food/water/predators impacting pest populations and activity. Yet an agent persisting in an environment will be able to directly observe how the state changes, providing time series data. Environments are often driven by complex processes, such as traffic being the product of actions taken by many individuals. This approach is data-driven, avoiding the need to create complex models of the underlying physics. It is increasingly common for robotic controllers to learn

the dynamics of the platform from data, saving effort creating models by hand. Applying time series forecasting can be considered analogous for understanding the environment around the agent, rather than the agent itself.

2.3 Long-term Autonomy

The development of persistent robotic platforms has grown interest in long-term autonomy. The majority of the research has been on localization and mapping in dynamic environments. Many approaches filter out the dynamic objects in an environment to more accurately generate a map of the static world (Hahnel, Triebel, Burgard, & Thrun, 2003) (Dayoub & Duckett, 2008). More closely related to our approach are methods that explicitly model the dynamics of objects in the environment in an effort to improve mapping (Mitsou & Tzafestas, 2007) (Ambrus, Ekekrantz, Folkesson, & Jensfelt, 2015).

Using models of the environment's dynamics for a robot's behavior is less common. Path planning through a dynamic network, where the traversability of paths changes over time, has been demonstrated. One approach modeled the uncertainty using a POMDP, applying forward search to find an efficient solution (Marthi, 2013). A similar approach modeled the dynamic state of an edge in a graph as a stochastic process, solving for a path with minimum expected travel time using dynamic programming (Loibl, Meyer-Delius, & Pfaff, 2013).

The most similar work is on "frequency map enhancement" (FreMEn), an approach that represents time-varying states as a binary variable, whose probability over time is modeled as a Fourier series. This predictive capability was applied in a variety of contexts, including localization (Krajnik T. , Fentanes, Santos, & Duckett, 2017), path planning (Fentanes, Lacerda, Krajnık, Hawes, & Hanheide, 2015), and information-driven exploration (Santos, Krajnik, Fentanes, & Duckett, 2016). The simplicity of the model limits the contexts to which it can apply, but makes the approach very efficient. This allows

the model to be applied to each cell within a 3D occupancy grid, creating a spatiotemporal map. Recent extensions to this work developed a warped hypertime representation to better model and predict the spatiotemporal dynamics (Krajník , et al., 2019). The probability density function of relevant state variables over both space and time were approximated using gaussian mixture models. While the workspace of a robot is finite, time extends forever. To deal with this, major periodicities are identified and events projected into a looping dimension of time with length equal to the duration of a period. In this representation events in the morning of consecutive days are closer than events in the morning and afternoon of the same day, allowing for standard machine learning tools to learn the relations between these events due to their distance in the representation.

While not traditionally considered long-term autonomy, research on solar energy harvesting for wireless sensor networks has some interesting similarities with the approach in this proposal. For sensor nodes, the goal is to reach "energy neutral operation" so that the node can continue to operate for long durations (on the order of years). To achieve this, both predictive forecasts of future energy and reactive responses to the actual gathered energy can be used to adapt the behavior of the sensor node (Kansal, Hsu, Zahedi, & Srivastava, 2007). In this case, action is limited to adjusting the duty cycle the sensor runs at, changing the amount of monitoring performed and energy expended.

2.4 Markov Decisions Processes and Reinforcement Learning

The goal of reinforcement learning is for an autonomous agent to learn how to behave in an environment to maximize the reward they receive. Reinforcement learning is most commonly formulated as a Markov Decision Problem (MDP) where either the model of the environment or the rewards the agent will receive are unknown. MDPs are defined as a set of states $S = \{s\}$, actions $A = \{a\}$, transition probabilities between states $T(s, a, s')$,

and rewards for acting and reaching a specific state $R(s,\ a,\ s')$. The output of a reinforcement learning algorithm is a policy $\pi(s)$: a mapping from state to action that maximizes the expected reward. The standard MDP formulation has no way to represent a changing environment. Information about the environment is encoded in the transition and reward functions, which are assumed to be constant. A naïve solution is to enlarge the state space to directly include discrete time and set the transition probabilities to specify that an agent must always step forward in time. This severely increases computational requirements.

An early attempt to deal with these challenges was to treat time as a special continuous state by creating a Time-Dependent MDP (TMDP) (Boyan & Littman, 2001). The durations of actions are assumed to be probability density functions (pdf) dependent on time and could either be relative (i.e., it takes on average 10 minutes to drive) or absolute (i.e., the train will arrive on average at 10:03am). The policy becomes a mapping of state-time pairs to actions, where the reward, transition, and value function all become functions of time. To make the Bellman equations solvable, this new time-value function is represented as a piecewise linear function, and the duration distributions restricted to discrete PDFs.

Recent work takes an alternative approach, viewing the changing environment as a changing transition function T and reward function *R* (Lantao & Sukhatme, 2018). This better captures the fundamental differences between time and space in (e.g., agents have no control of their location in time, time is asymmetric and unidirectional). The transition and reward functions at some time *t* in the future can be estimated using forecasts of the environment. The key to solving this new formulation, dubbed Time-Varying MDPs

(TVMDPs), is to estimate the transition time between states. With known transition times and the agent's policy, the time a state is reached can be found, and the proper transition and reward function applied when applying value iteration is used to solve for the policy. This maintains the size of the state and action space but limits the provided policy to a single start time and continuous execution. Any deviations and the expected arrival times for states, which the transition function is dependent on, are no longer valid.

The framework of MDPs and reinforcement learning seems capable of representing the types of problems this dissertation is focused on, but the author is not aware of a tractable method to do so. Generally speaking, representing the environment and how it changes over time increases the state space significantly. The slow nature means applying reinforcement learning in real-time is also difficult, as an environment with a strong daily cycle provides one true trial every 24 hours. Model-based reinforcement learning could be leveraged to solve this challenge. Using a complete model of the system, forward simulations could provide training at a significantly accelerated pace. If the solved policy learns only the best policy from a specific state forward, the approach becomes similar to model predictive controllers, constantly optimizing the future policy based on the current state and simulated future. A key difference with the work presented in this dissertation is the assumption of a complete model of the entire environment and agent. Rather than simulating the entire agent and environment forward in time, the utility of a specific behavior given knowledge of one aspect of the environment in the future is estimated.

2.5 Model Predictive Control

The core idea behind model predictive control (MPC) is that leveraging a model of a system allows you to optimize control inputs over the predicted future outputs. Using the given model, an optimal control sequence is found up to some horizon in time. While the control sequence is found for several time steps, only the first step is executed. At the next time step the state is measured and the process repeated. Models never perfectly describe real systems, and the iterative nature of the approach ensures the controller updates to respond to deviations from the predicted trajectory. Using a time horizon strikes a balance between costly excessive planning into the future, and short-sighted action based only on the current state.

Model Predictive Control has had early and significant success in the control of industrial processes (Qin & Badgwell, 2003). Some of the key features of MPCs were the relative ease of solving online optimization (as opposed to deriving a close-loop control law), the flexibility of models that can be used, and the inclusion of constraints for inputs and outputs. The limitation of requiring significant computation for each time step was irrelevant in this domain as the processes being controlled were generally slower, with sample times in the seconds or even minutes.

Some of the earliest applications of MPC to robotics found that while it provided good solutions, the computation made it infeasible for real-time control (Essen & Nijmeijer, 2001). However, this limitation was quickly overcome as efficient formulations and techniques for various problems were found (along with the ever-increasing speed of computing). Trajectory generation for multiple flying robots was solved in (Shim, Kim, &

Sastry, 2003) by writing the dynamics and cost functions in a form that allowed gradient descent to be applied for optimization. For trajectory tracking with a differential drive platform, a complete analytic solution to the prediction and optimization problem was found, enabling a very efficient solution (Klančar & Škrjanc, 2007). An excellent summary of various methods to reduce computation is provided in (Wang & Boyd, 2009).

Originally model predictive control was applied to dynamic systems where the model can be explicitly written as a differential or difference equations. Model predictive control allows for other types of models to leveraged in a straight-forward manner, the only requirement is that it can be used to predict the future state given certain control inputs. A common example is the use of neural networks to learn the dynamics of the system (Draeger, Engell, & Ranke, 1995) (Gu & Hu, 2002) (Williams, et al., 2017). Neural networks are popular due to their ability to approximate any non-linear function. A time series forecasting neural network was applied to forecast the disturbance coming into a system, rather than the system itself (Doganis, Aggelogiannaki, & Sarimveis, 2008). This was a special case for a production-inventory system, shown below in Figure 2.3, where sales were the disturbance and the forecasted demand helped the system compensate for changes. As opposed to treating demand as random noise into the system.

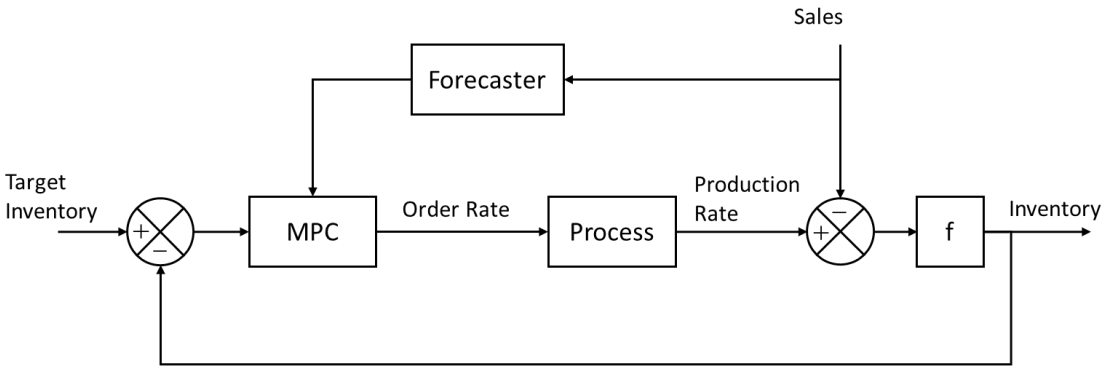

Figure 2.3: MPC controller leveraging forecasting from (Doganis, Aggelogiannaki, & Sarimveis, 2008).

The approach presented in this dissertation has similarities with model predictive control. Both use a model of the system (or part of it) to predict the future and adjust actions based on that prediction. As mentioned in the last section, the underlying assumptions in this work differ notably from classic MPC which leverages a complete model of the environment and agent. Chapter 3 will detail the formulation and assumptions used in this work.

CHAPTER 3. THE ARTIFICIAL CIRCADIAN SYSTEM

This chapter answers the first two subsidiary research questions: how the dynamics of the environment can be modeled, and how an agent can adapt its behavior using such knowledge. Before answering those questions, the term "dynamic environments" is defined to clarify the scope of this work. Then a concrete formulation of the entrainment problem, the problem of having an autonomous robotic agent entrain to a dynamic environment, is provided. Next the approach to modeling the environment, statistical time series analysis, is justified and the methods for applying it in real time are described. The core of this chapter will cover the action selection approach, i.e., how the agent uses forecasts from the time series model to adjust its behavior over time. Throughout the later sections a simple but detailed example will be used to demonstrate how the ideas can be applied in practice.

3.1 Dynamic Environments

This work defines a dynamic environment as one where properties of the environment change. The term "properties" is used to distinguish features of the entire environment from individual objects or agents within an environment. Consider the example an autonomous delivery robot. When navigating through empty roads it experiences a static environment. If traffic is added to this scenario, the robot must now deal with dynamic objects within its workspace, but the position of some specific car on the road would not be considered a property of the environment. An example of a property is the level of traffic, something of key interest

to anyone driving, which changes significantly over time. Traffic is not determined by any specific vehicle, but the interactions of many vehicles distributed within the road network. Traffic will greatly affect the travel time and fuel required to get somewhere, which means that knowing the level of traffic and how it is expected to change can help inform decisions on when and where to drive. It is these types of distributed properties of an environment that this dissertation considers.

The distributed nature of environmental properties makes modeling them more challenging. Continuing the traffic analogy, an autonomous car can see a nearby vehicle and immediately determine its position and velocity, which provides a strong predictor for where that vehicle will be in the near future. In comparison, there is no way for an individual car to determine how traffic will change given only the current amount of traffic. For the sort of environmental properties of interest, the true underlying state and dynamics are often complex and difficult to measure. In the delivery robot example, traffic is caused by the interaction of thousands of individuals. Another example is the activity of pests in a field which is generated by the interaction of the pest population with the local ecosystem. However, change in the property over time can follow trends and patterns. Time series models provide a structured way to identify and leverage the patterns from historical data.

The existence of patterns within these environmental dynamics is required for modeling to be meaningful. If environmental change is purely random, there is no way to predict the future, and a reactive approach to change is the best an agent can do. On the opposite side, if the environment is perfectly deterministic, then the exact state of the environment in the future is known, and optimal actions can be determined off-line and

provided as a schedule of behavior. Change in both natural and man-made environments almost always falls in between, neither completely random nor changing in a perfectly predictable way. Both clear patterns and significant random variation exist. In these cases, a forecasting system which continuously updates as new measurements are made provides the best predictions for an autonomous agent.

3.2 The Entrainment Problem

The primary research question motivating this dissertation is: *"How can the long-term dynamics of an environment be modeled, predicted, and exploited by a slow and persistent autonomous robot?"* The previous section clarified what "long-term dynamics of an environment" means in the context of this work. This section provides a concrete formulation for the problem of having an autonomous agent model and exploit those dynamics. The inspirational vision behind this work is an autonomous robot that can inhabit a complex environment and entrain its activity to the dynamics within it. This "Entrainment problem" is defined as:

1. A set of states, S, that represents both the agent's internal state, as well as relevant features of the environment it inhabits.
2. A subset, $\hat{S} \subset S$, of forecastable dynamic environmental properties. These states change over time regardless of the agent's actions, and are able to be predicted with some degree of accuracy. The agent may or may not be able to influence these states.
3. A set of agent behaviors, B, whose performance may depend on a subset of the forecastable dynamic environmental properties, $\hat{S}$.

4. For each behavior, a dependency function, $D(\hat{S}_t)$, that provides the expected utility of executing a behavior at a timestep given $\hat{S}$, where $D: \hat{S}_t \rightarrow X \mid X \in \mathbb{R}_{\geq}$

It is important to note that a complete model of the agent-environment dynamics is not given. Instead, the assumption is that realistic environments will have complex features changing due to causes that a robotic platform cannot measure. $\hat{S}$ is a subset of those features that have patterns that can be predicted, as discussed previously in Section 3.1.

The agent has some goals it is attempting to accomplish, for which progress can be represented with a notion of utility. $D(\hat{S}_t)$ captures how $\hat{S}$ impacts the result of a behavior, changing the expected utility. The true utility may depend on other features in S, but given that the agent cannot predict the value of those features in future, it cannot predict the utility of the executing the behavior in the future based on them. The assumption used is that increasing $D(\hat{S}_t)$ will increase the real utility and the performance of the agent. This assumption is likely to hold true when the forecastable property is a major factor for the overall performance of the behavior, and is not correlated with other non-forecastable features in the environment. Features of the environment that are both forecastable, and predictive of agent performance, are needed for the artificial circadian system to be useful.

3.3 Time series Forecasting

3.3.1 Approach

Foundational to this work is the answer to the first subsidiary research question, *"How can the long-term dynamics of an agent's environment be characterized or modeled?"* Without

an effective approach to understanding how the environment will change, a robotic agent cannot hope to adapt its behavior to those dynamics.

As circadian systems in nature were the original inspiration for this work, one might consider mimicking them as a first approach. This could be reasonable for environments with purely cyclic dynamics. An example of a similar model is provided in (Krajnik T. , Fentanes, Santos, & Duckett, 2017), where the environmental dynamics are modeled using Fourier series. Cyclic dynamics are of significant interest due to their prevalence and predictability. However, there are more general time series forecasting approaches that can both accurately capture cyclic dynamics and simultaneously model and predict non-cyclic effects.

In this work, we consider relevant states from the environment as time series. A variety of models and techniques have been developed in the field of time series forecasting, which focuses on the prediction of future values of some variable based on historical data. Along with capturing cyclic dynamics, usually referred to as "seasonal" effects in the time series literature, these approaches can model slow-moving trends and other statistical correlations between past and future measurements. A strength of time series modeling is that it is a data-driven approach. The underlying physics or domain knowledge about what in the environment creates the dynamics is not necessary. There is no need for expert knowledge. Instead, only a history of the state is required for model selection and parameter fitting. This simplifies applying the proposed architecture to new environments. Chapter 5 addresses how an agent can respond if this state history stops being a good predictor of future values (e.g., due to random disruptions or changes to the environment itself).

Statistical time series techniques, in contrast to machine learning based methods, are leveraged specifically. The use of machine learning in many fields is increasing, and time series forecasting is not an exception. Given the prevalence of ML within robotics, it may seem to be the natural choice. While ML based models may have the potential to provide superior forecasting, as discussed in chapter 2, they have yet to surpass even simple statistical approaches despite greater computational requirements (Makridakis, Spiliotis, & Assimakopoulos, 2018). The literature on statistical time series modeling and forecasting is more mature, providing libraries of standard approaches (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models, 2019). As the inputs and outputs of either type of trained model are the same, ML methods can be swapped in with minimal changes to the approach used in this dissertation if they show superior forecasting for a specific scenario.

The first step to forecast a relevant environmental variable is to choose an appropriate model. Section 2.2 provides resources with more detail, but the basic approach used here starts with an offline analysis of historical data to find what type of dynamics are present in the time series. Time-varying components (namely trend and seasonality) need to be identified and modeled. This includes finding the period of seasonality and whether these components are additive or multiplicative with the time series level determined (Hyndman R. , Koehler, Ord, & Snyder, 2008). These components can then be removed, leaving a time series that is not time-varying, also known as a stationary time series. Correlations within the stationary series should be represented with an ARMA model (Box, Jenkins, & Reinsel, 2008). When the forecasting error, or the difference between a forecasted point and its measured value, is a random series, then all the dynamics in the

time series have been captured. If there is a pattern within the error, there are unmodeled dynamics. Information criteria can be used to balance training accuracy with model complexity, aiding model selection and helping to avoid over-fitting. However, automated methods are still limited, and this process is typically done by hand. Once a model is selected, reliable methods for parameter estimation can be leveraged. In this work the R forecasting library (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models, 2019) is used for analysis of time series, parameter estimation, and forecasting.

In traditional time series forecasting the process is offline and data provided is grouped into a relevant interval (hours, days, months, etc). For application on robotic platforms, data collection is real-time, and the data must be discretized by choosing an appropriate interval length. Shorter intervals provide higher resolution, but less data per interval to use. Forecasting accuracy is also heavily influenced by the number of steps forward in time being predicted. Part of the offline analysis process requires choosing an appropriate interval length that balances the resolution an agent needs to see relevant changes in its environment with the accuracy of its forecasts.

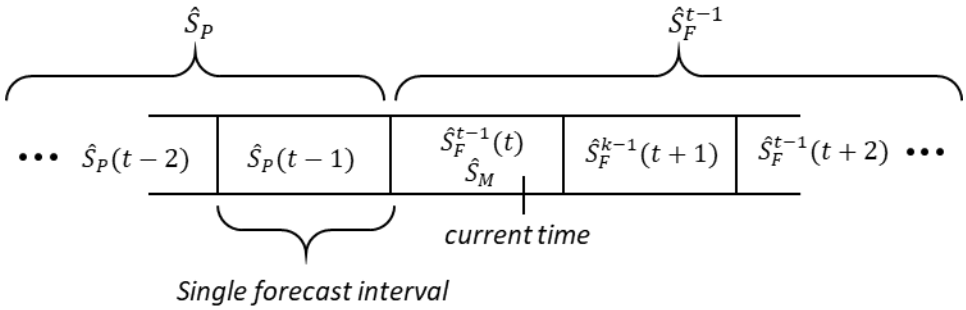

Figure 3.1: The representation of the forecastable state over time, broken into the past history (S_P), present measured (S_M), and future forecasted (S_F).

Given a time series model, the agent uses three data structures to update the forecast with real time sensing. The past state, or the measured time series, is stored as the vector $\hat{S}_P$. These are fed into the time series model and used to forecast the future values of the state, $\hat{S}_F^t$, where t is the timestep during which this forecast vector was generated. Lastly, there is a set of current sensor measurements called $\hat{S}_M$. These measurements could be anything from individual events that are being counted to separate measurements that will be averaged together. At the end of a forecast interval, $\hat{S}_M$ is used to generate a final state value for that interval. This process is described by the algorithm in Table 3.1. The example in section 3.5 will show how a specific time series model can be updated to fit to new data and used to forecast.

Table 3.1: Algorithm to apply discrete time series models in real time on autonomous agents.

Algorithm: Forecasting Loop	
Data structures:	$\hat{S}_P$ – Vector of past measurements of relevant state $\hat{S}_M$ – Vector of current measurements of relevant state $\hat{S}_F$ – Vector of future forecasts of relevant state TS – Time series model object with functions fit and forecast k – Current timestep
Functions:	$Percept(X)$ – Converts the measurements X to a single estimate for one time-step $TS.Fit(X)$ – Fits the TS model parameters to the data vector X $TS.Forecast(X)$ – Forecasts the future values of time series X using model
Initialization: $\hat{S}_P \leftarrow historical\ data\ for\ initialization\ of\ model$ $TS.fit(S_P)$ $\hat{S}_F \leftarrow TS.forecast(\hat{S}_P)$ $t \leftarrow 0$ **Repeat:** $t \leftarrow t + 1$ For duration of time interval: $\hat{S}_M \leftarrow relevant\ sensor\ input$ $\hat{S}_P.append(Percept(\hat{S}_M))$ $\hat{S}_F \leftarrow TS.forecast(\hat{S}_P)$	

The main application of such a model is to forecast the future state, but there is a close relation to the more general ideas of state prediction, filtering, and smoothing. Forecasting is prediction, and much of this dissertation focuses on how it can aid an agent in making decisions based on the future state. Forecasts from past data can also be used when estimating a current state that the robot has not or cannot measure. This idea can be leveraged in cases with limited sensing and will be discussed more in Chapter 4. If a robotic agent's sensing is particularly noisy, filtering and smoothing may also be useful. It is possible to develop true smoothing and filtering operations within some time series modelling frameworks (Hyndman R. , Koehler, Ord, & Snyder, 2008).

A notion of entrainment to the environment is captured in the system at two levels. Every forecast interval the agent appends a new value to $\hat{S}_P$ and generates a new forecast vector $\hat{S}_F^k$. This means the forecast for some specific point in time in the future is repeatedly updated (usually becoming more accurate) as new measurements of the environment are taken. In this context, entrainment can be viewed as a feedback loop, continuously adjusting the forecasts (and therefore the behavior of the agent) as new measurements come in. Secondarily, the new history, $\hat{S}_P$, can be used to re-estimate the model parameters adapting the time series model if the dynamics of the environment have changed. This latter context is more akin to learning, and Chapter 5 will explore this idea in an effort to make the approach robust to changing dynamics.

This section discussed how time series modeling and forecasting is a data-driven approach. This is most useful when a model of the underlying process is either not available or not reasonable to use. As section 3.1 stated, many environmental properties that could

impact an autonomous robotic agent are like this, with traffic being the most clear and common example. Time series models fit well with common cases considered in this dissertation, which focuses on the idea of a persistent robot in a complex natural environment. However, when physics-based models are available, they may provide superior predictions and should be considered. Note that for the rest of the work, the type of model does not matter, as long as the model can produce predictions of the future state.

3.3.2 Forecasting Example Application

Here an example is introduced to demonstrate the application of the approach in a simplified scenario. This will be a running example, used through the following sections in this chapter. Consider an autonomous electric vehicle that has several tasks it is used for, including a daily delivery of supplies to locations across the city it is based in. The vehicle can drive for a few hours on one charge, and traffic will impact how much useful work it can perform before having to stop and recharge. How can the artificial circadian system be applied to time the deliveries to the traffic patterns of the city? The first step, described in this section, is to forecast the traffic.

Publicly available traffic data from Arlington, VA (Bike Arlington, 2020) will be utilized. Five days of the data of are shown below in Figure 3.2. While this data measures traffic counts at a specific location, it will be used in this example as an overall level of traffic throughout the city.

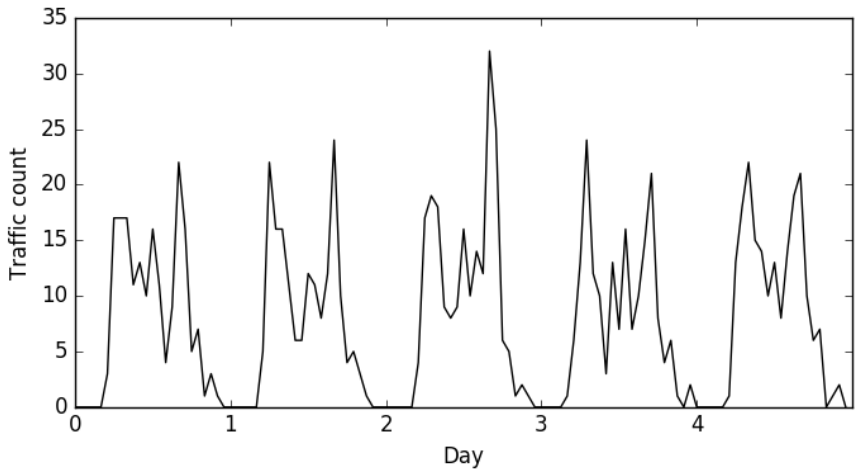

Figure 3.2. Traffic data used for this example.

The data in Figure 3.2 shows clearly cyclic dynamics (referred to as "seasonal" in the time series literature) but no obvious trend. Let us apply one of the simpler time series models. From (Hyndman R. , Koehler, Ord, & Snyder, 2008), the basic exponential smoothing model for a time series with only additive error and seasonal components is:

$$\hat{y}_{t+h|t} = l_t + s_{t+h} \tag{3.1}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)l_{t-1} \tag{3.2}$$

$$s_t = \gamma(y_t - l_t) + (1-\gamma)s_{t-m} \tag{3.3}$$

Where h is the number of time steps past the current time t, m is the period of seasonality, α and γ are smoothing parameters, l is the current level of the series, and s is a vector holding an average value for each timestep within a cycle. Equation 3.1 is the forecasting equation, which uses the level and seasonal components to provide the forecast h steps into the future. Equations 3.2 and 3.3 are used to update the level and seasonal components based on new measurements y_t.

The parameters for this model are the smoothing parameters α and γ, weights between 0 and 1 that determine how much a component changes give an update, and the initial states of components l and s. The R forecasting library is used to find the smoothing parameters and initial values for the level and seasonal terms by applying likelihood maximization to the historical data. See (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models, 2019) for details on the R forecasting library. The command used and output model parameters are shown below:

```
Command:
  model_fit <- ets(traffic_history, model='ANA')

Model details:
  α = 0.1613
  γ = 0.0001
  l = 7.6273
  s_m = {-8.1579, -7.204, -6.0761, -6.1372, -3.3255, -0.3216,
  8.0552, 13.6075, 6.5137, 2.2273, 3.4802, 2.9286, 3.5316,
  3.1464, 4.803, 7.8877, 9.4509, 8.7658, -4.9698, -7.4824, -
  7.525, -7.458, -7.8846, -7.8558}
```

Consider if we had the model and parameters given above. Initially, the forecast for three timesteps into the future is $\hat{y}_{t+3|t} = l_t + s_{t+h} = 7.6273 + (-6.1372) = 1.4901$. This makes sense as that is still early in the morning, a time when there is historically very little traffic. Now consider the agent receives several updates with higher than average traffic for the time. Remember that y_t is the measured value at timestep t, while $\hat{y}_{t|t-1}$ is the forecasted value given the data at time $t-1$. First consider a new measurement comes in at time $t+1$:

$$y_{t+1} = 5.0$$

This can be put into equations 3.2 and 3.3 to update the level and seasonal components:

$$\begin{aligned} l_{t+1} &= \alpha(y_{t+1} - s_{t+1-24}) + (1-\alpha)l_t \\ &= (0.1613)(5.0 + 7.204) + (0.8387)(7.6273) = 8.366 \\ s_{t+1} &= \gamma(y_{t+1} - l_{t+1}) + (1-\gamma)s_{t+1-24} \\ &= (0.0001)(5.0 - 8.366) + (0.999)(-7.204) = -7.197 \end{aligned}$$

The new components can be used in equation 3.1 to produce new forecasts of the future time-steps:

$$\begin{aligned} \hat{y}_{t+2|t+1} &= l_{t+1} + s_{t+2} = 8.366 + (-6.0761) = 2.29 \\ \hat{y}_{t+3|t+1} &= l_{t+1} + s_{t+3} = 8.366 + (-6.1372) = 2.229 \end{aligned}$$

This process can be repeated when new data comes in at time $t + 2$:

$$\begin{aligned} y_{t+2} &= 7.0 \\ l_{t+2} &= \alpha(y_{t+2} - s_{t+2-24}) + (1-\alpha)l_{t+1} \\ &= (0.1613)(7.0 + 6.0761) + (0.8387)(8.366) = 9.126 \\ s_{t+2} &= \gamma(y_{t+2} - l_{t+2}) + (1-\gamma)s_{t+2-24} \\ &= (0.0001)(7.0 - 9.126) + (0.999)(-6.0761) = -6.07 \\ \hat{y}_{t+3|t+2} &= l_{t+2} + s_{t+3} = 9.126 + (-6.1372) = 2.989 \end{aligned}$$

We see the seasonal component changes little due to the very low value of γ. This implies there is a strong historical average that seems persistent through random perturbations. However, the value of the level component shifts up, increasing the forecasted amount of upcoming traffic as the agent gets more data. Figure 3.3 below shows the application of these update and forecasting equations to the historical traffic data for two days of forecasts. Note that for this simple model the forecast will repeat every cycle going forward. This forecast will be fed into the action selection system, changing the activation of relevant behaviors.

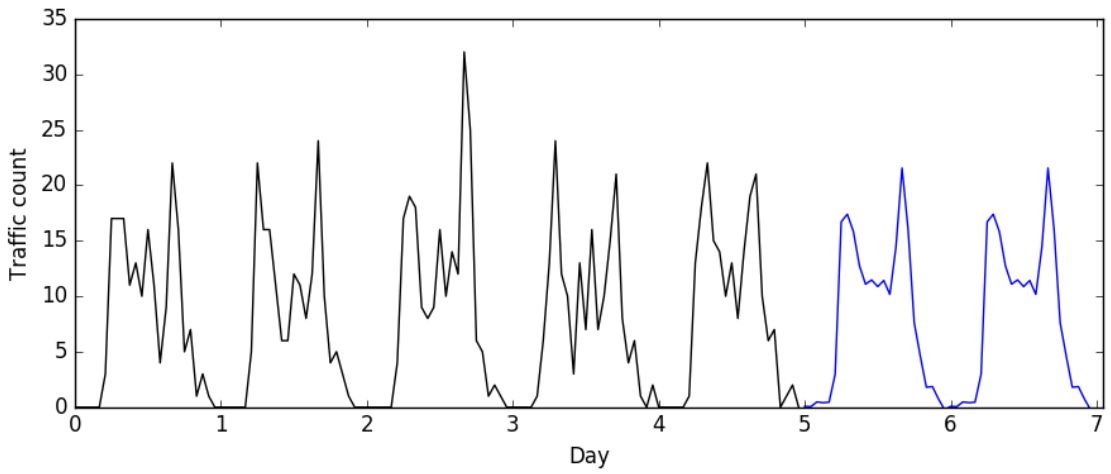

Figure 3.3: Traffic data (black) plus two days of forecasted traffic (blue).

3.4 Action Selection

Given forecasts of how the environment will change and knowledge on how the environment impacts performance, how can a robotic agent adapt its behavior? This is the second subsidiary research question and the focus of this section. The core action-selection mechanism is ethologically inspired and based on activation levels. First, a circadian function that turns forecasts into activation for a behavior is constructed. Then, how this circadian function is integrated into the action-selection approach is described.

3.4.1 The Circadian Function

The circadian function takes the forecasted state ($\hat{S}_F$) and provides activation for a behavior. It is designed to leverage three qualities of the targeted domain of slow, persistent agents:

- For slow and persistent agents, behaviors will often be executed for long durations of time

- The environment may change significantly during the execution of a behavior
- The environmental dynamics are dominated by a cycle with known period

The first two points imply that any instantaneous measurement or prediction of the environment may be inadequate. Instead, we need to consider the state of the environment over the expected duration of the behavior. In other words, we are concerned with the agent's activity over time. The last point is a convenient property of the types of dynamics we are most interested in.

As discussed in Section 3.2, $D(\hat{S})$ encodes the dependency of a behavior's performance on the subset of forecasted states. $D(\hat{S})$ is mostly unconstrained and may be any function that outputs non-negative real numbers ($D: \hat{S} \rightarrow X \mid X \in \mathbb{R}_{\geq}$). In many common cases, performance will be roughly proportional to key environmental properties (e.g., power collected increasing linearly with solar irradiance). Another common trait may be a minimum/maximum or saturation point, after which the state may increase but not impact performance. With a forecast of the state in the future and an estimate of the robot's performance given that state, the last element needed is the time an agent would be executing the behavior. Specifically, delays before starting a behavior's useful work (generally due to needing to move to a location) noted as t_d, and the total execution time, noted as t_e, are used. Both variables can take calculated values if available (e.g. known distance to a goal location and known travel speed) or average values from previous executions otherwise. With these components, we can estimate the period of time a behavior will be executing and the total performance:

$$P(\hat{S}, t, t_d, t_e) = \sum_{\tau=t+t_d}^{t+t_d+t_e} D\left(\hat{S}(\tau)\right) \tag{3.4}$$

Equation 3.4 shows the predicted performance function, P, which outputs the performance for the full execution of the associated behavior starting at time t given the forecasted state, delay, and execution duration. As a circadian function is behavior-specific, the fundamental question is always *"should this behavior activate now?"* This is answered by comparing the expected performance of executing now versus in the future. To do so, P is applied as a sliding window, shown in Figure 3.4, giving the performance of executing the behavior over a range of times into the future.

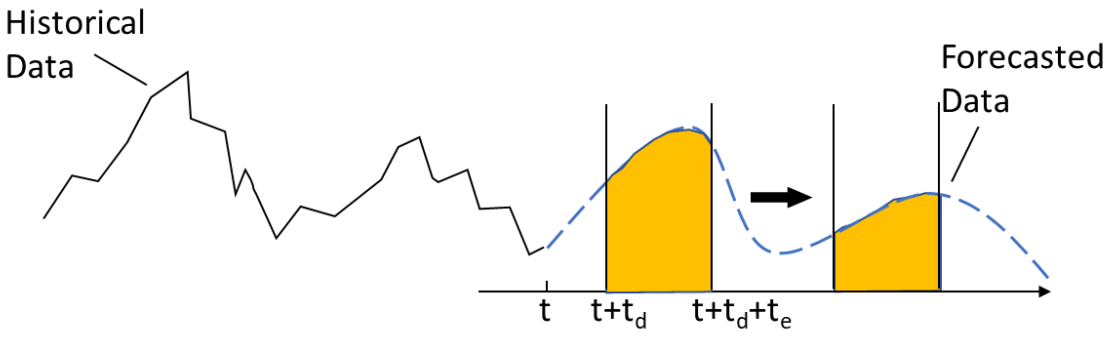

Figure 3.4: Visualization of the predicted performance function, P, which acts as a sliding window over the forecasted state, determining the performance for executing a behavior at a certain time t.

This introduces the question of how far into the future should the estimated performance be calculated? The circadian function is concerned with relative performance within one cycle, rather than the absolute level of performance. For example, if the agent gets a second solar panel and can collect twice as much energy from the same amount of sunlight, the best hour of the day to charge is unchanged. Note that if the agent needs less total time to charge, the duration of the behavior execution (t_e) will change, impacting the performance

for executing at a specific time. The final step for the circadian function is to find the average and maximum values for a behavior's performance over one cycle and compare that to the performance for executing now. The heuristic used will provide activation when it is better than average to execute, with maximum activation at the best time. In the following equations, H is the period of the cycle.

$$P_{avg} = \frac{1}{H}\sum_{\tau=t}^{t+H} P(\tau) \tag{3.5}$$

$$P_{max} = \max\{P(\tau) \mid \tau = t \ldots t + H\} \tag{3.6}$$

$$CIR(t) = \left(\frac{P(t) - P_{avg}}{P_{max} - P_{avg}}\right)^{n} \text{ if } \mathrm{P(t)} > \mathrm{P_{avg}}, \text{ else } \mathrm{CIR} = 0 \tag{3.7}$$

A naïve approach may provide full activation at the maximum time-step and no activation anywhere else. If no other factors impacted the ability for the agent to execute the behavior, this would be the best strategy. In practice, it is expected that the agent will have other behaviors and constraints and will not be able to execute a behavior at any given time. Providing activation over a range of times ensures the behavior will have multiple opportunities to execute. The parameter n in equation 3.7 allows one to select how much to bias towards the max value. Figure 3.5 below shows how higher values decrease activation for sub-maximal performance. In practice, $n = 1$ is expected to be effective in most cases.

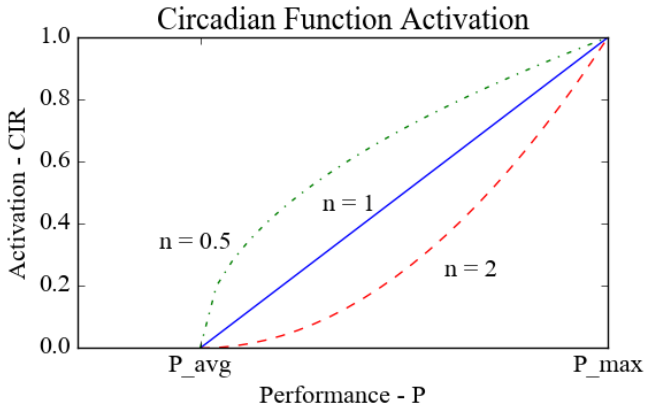

Figure 3.5: The parameter n adjusts the bias of the circadian function towards the maximum performance, with higher values of n making a sub-maximal performance provide less activation.

3.4.2 Circadian Function Example Application

This section continues the example of an autonomous delivery car dealing with traffic. Section 3.3.2 detailed how forecasts of the traffic into the future are generated. Here those forecasts will be used within the circadian function. The circadian function also requires the dependency of the behavior's performance on the state. Let us assume for this example that an increase in traffic proportionally increases the duration needed to travel some distance according to a parameter k, with a known minimum duration when traffic gets low enough (min_dur). There is a maximum performance (max_perf) value that decreases as the duration to travel increases. The performance for a given timestep with a level of traffic S, is:

$$D(S) = max\_perf - \max\{kS, min\_dur\} \tag{3.8}$$

For this example, some arbitrary values are used: max_perf = 7.5, k = 3/10, and min_dur = 3. Let us assume the vehicle can operator for three hours ($t_e = 2$) and there is

one hour of setup time before it can begin ($t_d = 1$). Plugging the forecasted traffic values from section 3.3.2 into this equation, the estimated utility at each time step is found. The estimate utility for timesteps 9-11, with forecasted traffic values of $\hat{S}[9,11] = \{12.7, 11.1, 11.5\}$, are:

$$D\big(\hat{S}(9)\big) = 7.5 - \max\left\{\frac{3}{10}(12.7), 3\right\} = 3.68, \qquad D\left(\hat{S}(10)\right) = 4.18,$$

$$D\left(\hat{S}(11)\right) = 4.06$$

These timesteps are when the behavior activating at t=8 is expected to execute. As described in equation 3.4, the predicted performance for activating the behavior at t=8 is sum of the utility over the expected duration of the behavior:

$$P\big(\hat{S}, 8,1,2\big) = \sum_{\tau=8+1=9}^{8+1+2=11} D\left(\hat{S}(\tau)\right) = 3.68 + 4.14 + 4.06 = 11.88$$

Repeating this calculating at every timestep in the cycle allows us to find the max and mean value for the predicted performance. In this example, they are $P_{max} = 13.5$, and $P_{avg} = 11.64$. The final circadian activation for our example timestep t=8 is given by equation 3.7:

$$CIR(t = 8) = \left(\frac{\mathrm{P}(\hat{\mathrm{S}}, 8) - \mathrm{P_{avg}}}{\mathrm{P_{max}} - \mathrm{P_{avg}}}\right) = \left(\frac{11.88 - 11.64}{13.5 - 11.64}\right) = 0.129$$

The circadian activation at this timestep is labeled below in Figure 3.6, which also shows the predicted performance and circadian activation at any given time in the cycle, generated using the forecast from section 3.3.2. There are regions of high activation

during the middle of the night, when traffic is very low. There is also a range of time during the middle of the day where performance will be decent, e.g. traveling will not take too long.

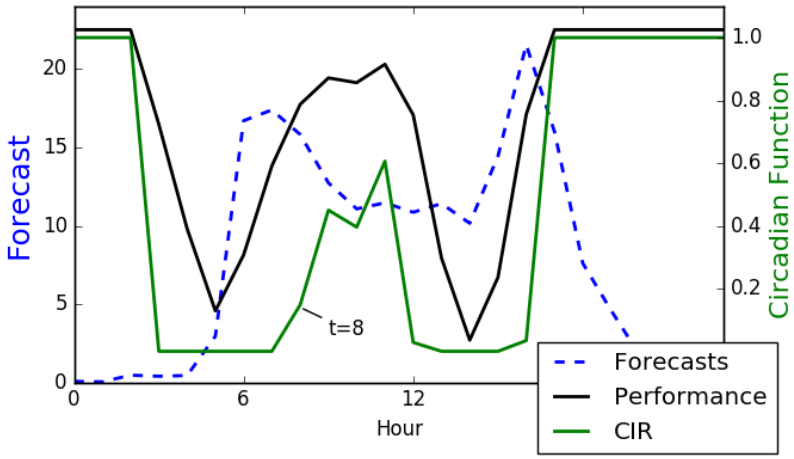

Figure 3.6: Forecasted state, predicted performance, and circadian activation for one environmental cycle into the future. The scale for predicted performance P is not shown, but is aligned against the circadian function for easier comparison.

3.4.3 Behavior Activation

The action-selection approach is primarily based on the method described in (Arkin, Fujita, Takagi, & Hasegawa, 2003), but draws from a rich set of work in this area (Chernova & Arkin, 2007) (Blumberg, 1994) (McFarland & Bösser, 1993). The core idea is simple: each behavior has an activation function, and the behavior with the highest activation is executed. Activation functions are broken into two components: the motivation function and the releasing mechanism. The motivation function (MOT) provides activation based on internal, or endogenous, variables that represent the state of the agent and its goals. The releasing mechanism (RM) provides activation based on external, or exogenous, variables which represent the state of the environment around it. An activation function is defined as:

$$Act = G * MOT * RM \tag{3.9}$$

A useful interpretation is that the releasing mechanism represents the effectiveness of executing a behavior based on the environmental state (e.g., higher solar irradiance means more power from charging). The motivation function provides how much the agent values the expected change (e.g. a lower battery means charging some amount is more valuable). The final component, G, is the behavior gain: a scalar weight used to set the priority between different behaviors.

The motivation function and releasing mechanism themselves are both defined as a weighted summation of functions of relevant variables (e.g. battery charge, distance to or position of a goal, traffic level, etc). Each component function describes how much the behavior activates based on the relevant variable's value, outputting between 0 and 1. Thus, the MOT and RM also return between 0 and 1. In (7) below, v(i,j) are relevant variables, f(i,j) are the component functions, and w(i,j) are the weights of each component.

$$MOT = \sum_i w_i * f_i(v_i) \qquad RM = \sum_j w_j * f_j(v_j) \tag{3.10}$$

The circadian function (CIR) provides activation at the best times to execute over a cycle based on a forecasted environmental variable. A key observation is that the circadian function should impact the activation generated by the environmental variable being forecasted. If part of a releasing mechanism is a function of some variable, $f_j(v_j)$, then that would be replaced with a weighted sum of this original component and the circadian function:

$$f_j(v_j) \;\rightarrow\; w_c CIR(t) + (1 - w_c) f_j(v_j) \quad 0 \leq w_c \leq 1 \tag{3.11}$$

The parameter w_C allows adjustment from not using the circadian function at all (w_C=0) to relying on it completely (w_C=1). In practice, forecasts are never guaranteed to be right, and it is useful to consider the current measured value of some environmental variable in the computation. Later, in chapter 5, we will look at how making w_c a function of forecast accuracy can allow the action-selection system to adapt online to changing dynamics and better deal with inaccurate forecasts.

A final consideration is the issue of behavior dithering. When activation levels change smoothly a rapid switching of behaviors can occur which degrades overall performance. Consider an agent that continuously leaves and returns to a charger, as it collects or expends just enough energy for a charge behavior's activation to cross the threshold of "highest activation". Some method of prioritizing the active behavior to create behavioral persistence is needed. The most common approaches are inhibition of inactive behaviors by the active behavior (Blumberg, 1990), or hysteresis in the action selection process (Velayudhan & Arkin, 2017). This approach uses the latter, and each activation function is supplemented with a hysteresis function that defines a minimum amount of time the behavior will execute.

The approach described in this section is sufficiently general in that it is compatible with many action selection methods. If a relevant environmental variable is represented as a scalar value and used to calculate some component (activation, utility, cost, etc.) then the circadian function can be substituted in, extending the method to consider the best time to activate in the dominant cycle.

3.4.4 Behavior Activation Example Application

Continuing the example implementation, the behavior to deliver goods will not activate exclusively based on the output of circadian function. Let us consider if there is a releasing mechanism comparing the current level of traffic, which we will assume is perfectly known, to the maximum traffic (estimated to be 30). There is also a motivation function based on the time since last delivery which maximizes internal motivation after 48 hours without a delivery.

$$MOT = \frac{time\_since\_last}{max\_time} \tag{3.12}$$

$$RM = 1 - \frac{S}{traffic\_max} \tag{3.13}$$

Using equation 3.11 to include the circadian function into the RM, and inserting both the MOT and RM components into equation 3.9, we get the equation for the total activation of the behavior shown in equation 3.14. The activation for this example is plotted below in Figure 3.7 against the forecasted traffic levels. In this example $w_c = 0.8$ was used, and the behavior gain G is set to 1.

$$Act(t,S) = MOT(t) * \big((1 - w_c) * RM(S) + w_c * CIR(t,S)\big) \tag{3.14}$$

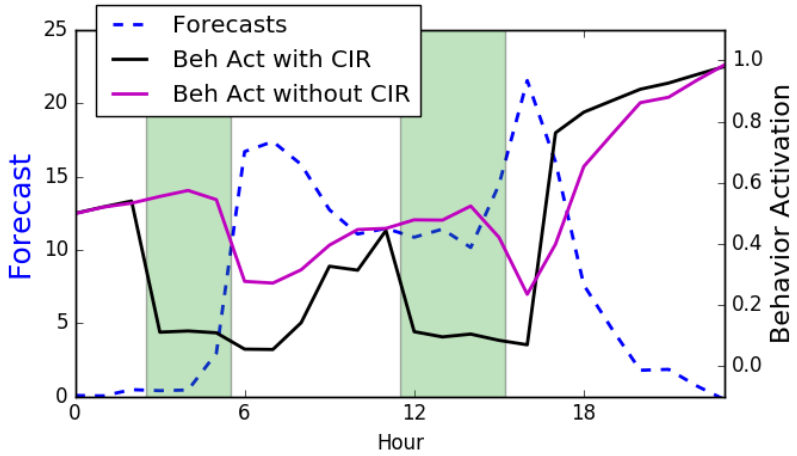

Figure 3.7: The total activation of the delivery behavior when using the circadian function (black) and when not using the circadian function (purple). Generated by applying equation 3.14, with *time_since_last* starting at 24 hours, and the real traffic S for a time step equal to the forecasted traffic. The green shaded regions mark areas where the circadian function significantly changes the total behavior activation.

Figure 3.7 compares the delivery behavior's activation with and without the circadian function. When the level of traffic is unchanging, these reduce to nearly identical activation as the current traffic is the future traffic. Two regions of large deviation are marked with green shading. These are areas where traffic is low but will soon increase, and activation of the "delivery" behavior would cause the agent to end up stuck in bad traffic. This is one of the simplest uses of the artificial circadian system. Allowing the agent to avoid activating at bad times due to informed predictions of the future, even when current measurements indicate it would be advantageous to activate.

3.5 Summary

This chapter answered the first two subsidiary research questions and in doing so developed an architecture for an autonomous robotic agent to entrain itself to dynamic environments. First, the term "dynamic environments" was defined to clarify the scope of this work, and a concrete formulation of the problem of entraining an agent to a dynamic environment

was provided. The chosen method for modelling the environmental dynamics, statistical time series, was justified, and its application in a real-time context on a robotic platform was detailed.

Using the forecasted state and the knowledge of how that state impacts a behavior's performance, a circadian function was developed that adjusts the activation of its associated behavior based on the future predicted state. The performance of a behavior over the dominant environmental cycle is estimated. The circadian function provides activation when it is better than average to execute the behavior, with max activation at the best time within a cycle.

Finally, this circadian function is incorporated into a general action-selection methodology. While a behavior's activation is dependent on multiple factors, the circadian function provides a strong bias to execute behaviors at the best times within an environmental cycle. A simple example detailed how this approach can be applied to an agent dealing with traffic. The more comprehensive experimental validation in chapter 4 will demonstrate how behavior entrained to the environment's dynamics can be produced.

CHAPTER 4. A NOTIONAL AGRICULTURAL EXPERIMENT

The artificial circadian system has been presented as an answer to the first and second subsidiary research questions. It is a method to model environment dynamics on an autonomous robotic agent and entrain its behavior to them. This chapter describes the experiment built to validate this proposed answer. It will entail an autonomous agent performing persistent, agricultural-themed tasks. Daily and seasonal environmental cycles will drive the changing presence of pests, weeds, and sunlight. Successful entrainment of the agent's behavior to its environment will be shown by quantitative improvements in performance and qualitative analysis of the scheduling of activity.

A second goal of this experiment is to answer the fourth subsidiary research question: "*In what contexts is entraining to a dynamic environment useful for a robotic agent?*". Chapter 3 discussed the type of dynamic environments that are relevant, but qualities of the agent itself have so far not been addressed. If an agent can sense and react to environmental change fast enough, there may be no need for any type of prediction or forecasting. Simple reactive responses to those changes may be enough. As discussed in chapter two, a reactive agent will still generate cyclic behavior in response to a cyclically changing environment. In the following experiment, specific scenarios tested both slow actuation and limited sensing. These scenarios impeded the agent enough to degrade performance in its task and tested if the artificial circadian system was able to compensate for those limitations. Dynamic weather based on historical solar measurements was also applied to test the ACS approach against realistic dynamics for a solar-powered robot.

This chapter first covers the construction of the agricultural testbed. Then, the full implementation of the experiment will be detailed, including the environmental dynamics, agent behaviors, scenarios, measures, and procedures. Next the results of each scenario will be presented, and the core hypotheses of the experiment answered. Finally, a framework for describing the scope of the ACS, an answer to the fourth subsidiary question, will be presented and the chapter will conclude.

4.1 Construction of the Agricultural Testbed

Validation of the artificial circadian system brings several challenges. This research is on the interaction of a robotic agent with a dynamic environment. Most robotic laboratories are not dynamic. The research is also focused on agents persisting over long durations of time. If a single trial requires multiple days, collecting multiple trials for multiple scenarios becomes an incredibly time intensive process. To make this work tractable, a notional agricultural environment was created for both simulation and hardware.

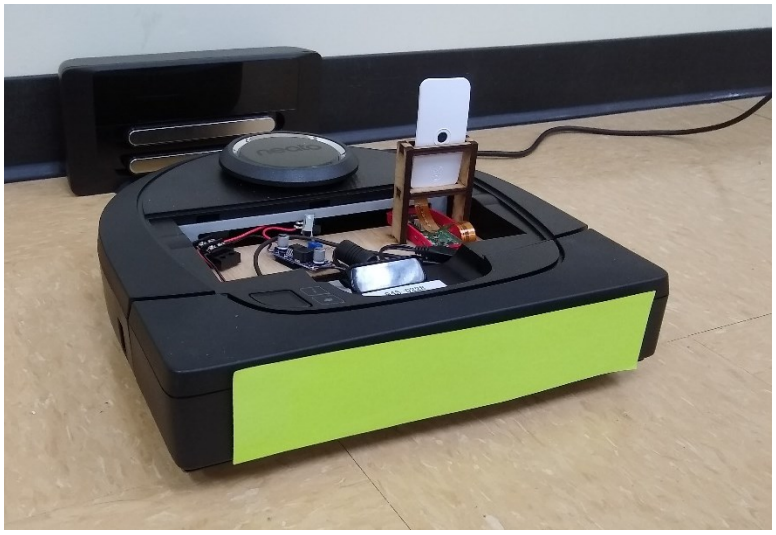

Figure 4.1: The Neato Botvac platform used in the physical experiment, with the Raspberry Pi controller and camera mounted. The charging station is shown behind it.

The agricultural testbed consists of a mobile robot, a charging station, and sixteen artificial plants. The mobile robot is a Neato Botvac (see Figure 4.1) which comes equipped with a 2D lidar for localization and the ability to dock to recharge (a key feature for the long duration experiments). The Botvac has been supplemented with a Raspberry Pi for control and a camera to detect the LED outputs of the artificial plants.

The core feature of the environment is the artificial plant, which uses three LEDs to display its state. Red indicates the presence of pests, green for weeds, and blue indicating a lack of water. Each artificial plant also has an IR range sensor, which acts as a switch the robot can interact with by moving close. A fourth white LED indicates when the robot is close enough to trigger the switch. Each of the eight cylinders shown below has two independent sides for a total of sixteen artificial plants in the testbed. A plant is "monitored" when the agent sits close enough to an artificial plant to trigger the distance sensor, essentially docking to it. To remove a pest the agent backs up from this docked position and then returns after a brief delay, toggling the distance switch OFF-ON. To remove a weed, this procedure is repeated, toggling the switch OFF-ON-OFF-ON.

Figure 4.2: The physical testbed. Each row shown has artificial plants on both sides. The green shrubs shown are decorative fakes, no plants were harmed in this work.

Each row of artificial plants, shown above in Figure 4.2 is controlled through an Arduino, connected to a base laptop. This laptop runs the environment model, generating pest and weed activity to display on the artificial plants, and updating the model when the agent acts to remove pests or weeds.

4.1.1 Simulated version

A simulated version of the agricultural testbed was critical to get more data in reasonable time frames. The simulation uses ROS/Gazebo. The physical size of testbed and speed of the robotic platform were measured to match simulation parameters to. Results detailed in section 4.3 showed overall good matching between hardware and simulated trials.

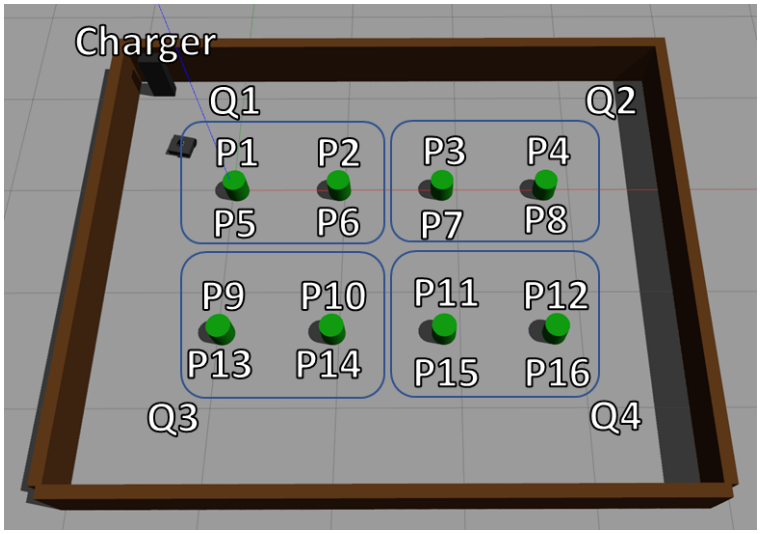

Figure 4.3: The gazebo simulation of the physical testbed. In this image, the locations of individual artificial plants (P1 through P16) and quadrants that share pest populations (Q1 through Q4) are labeled.

4.2 Experiment Design

4.2.1 Environmental Variables

The first dynamic environmental variable is solar power. The simulation of solar power is based on data from the National Solar Radiation Data Base (NSRDB) (National Solar Radiation Database, n.d.). Available solar radiation is primarily driven by the daily solar cycle, with random influence from weather. The chosen data uses measurements from near Atlanta in 2016. Figure 4.4 shows six consecutive days of solar irradiance. Days with clear skies have smooth curves, while days with cloudy weather have sporadic levels of available solar energy. Below this, Figure 4.5 displays the daily net values for the entire year. It shows the yearly cycle in daily solar irradiance, and a view of the prevalence of disruptions due to weather. To isolate the impacts of weather from other experimental variables, some scenarios use "static" solar dynamics, for which days with impact from weather are removed.

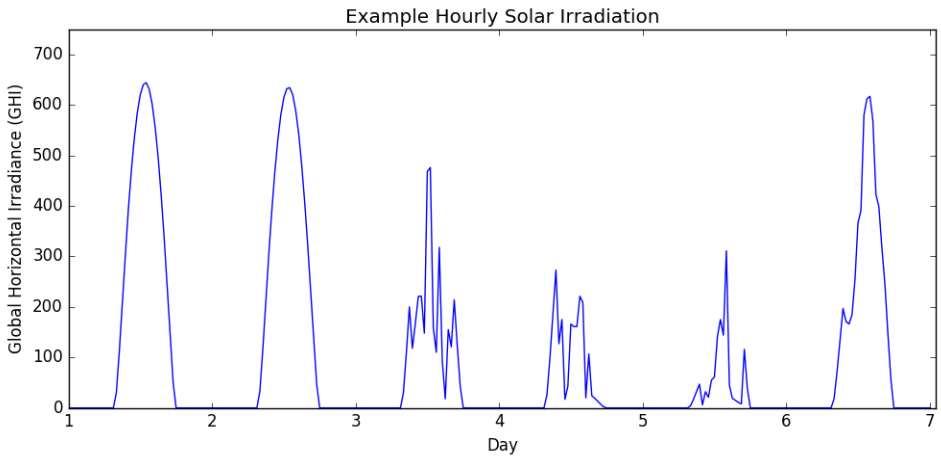

Figure 4.4: Solar irradiation data for several consecutive days. There are examples of clear and cloudy weather.

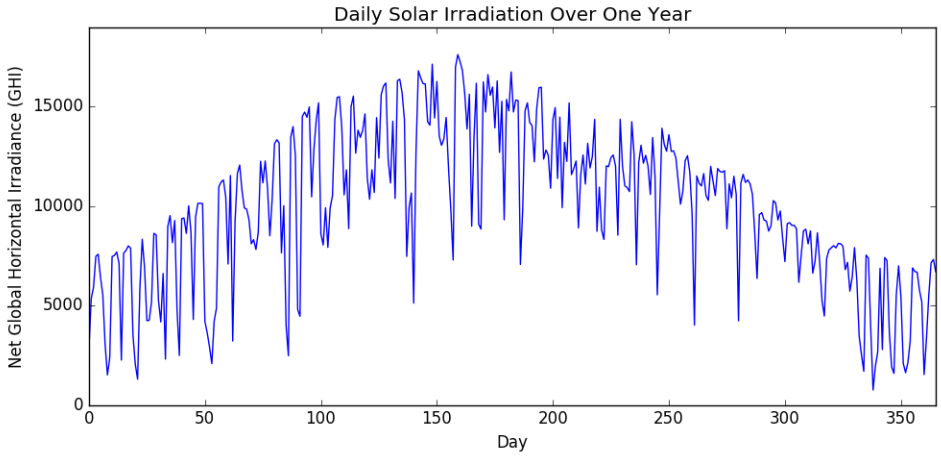

Figure 4.5: Solar irradiation data for a full year. The max follows a smooth curve based on the seasonal change, while weather causes regular but random decreases in available energy.

As weather forecasts are some of the most advanced and widely available forecasts, this experiment simulates the agent receiving weather forecasts from an outside entity. Direct forecasting of solar irradiance using time series is possible (Bacher, Madsen, & Nielsen, 2009) but inherently more limited due to the smaller set of data available on a single robotic platform. Forecasts are generated and provided to the agent by taking the ground truth and multiplying by a random factor. This factor increases exponentially the further into the

future the forecast is. Based on literature (Lorenz, Hurka, Heinemann, & Beyer, 2009) a relative RMSE of 30% for forecasting 24 hours into the future was chosen as a realistic estimate of error.

The second environmental variable, pest dynamics, is broken down into quadrants with the four plants within a quadrant sharing a single pest population. Pest population uses a simplified linear growth model. Based on the population, pest activity is generated. It is this activity that the agent measures and interacts with. The agent can only remove a pest when it is active at the plant being monitored. Many insects have daily cycles in activity. This experiment used a nocturnal pattern peaking in the night, with low activity (20% of max) at other times. Figure 4.6 shows an example of this pest activity over time, including agent intervention, for one quadrant.

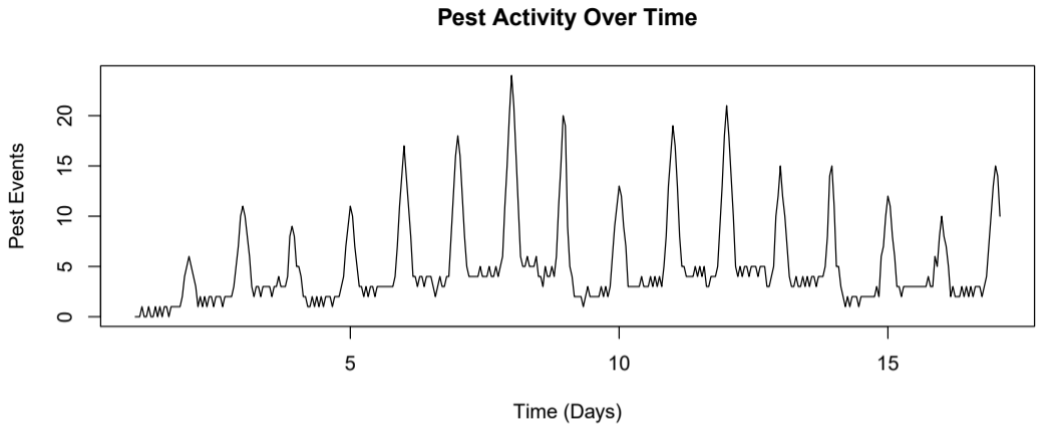

Figure 4.6: Pest activity for one quadrant over seventeen days. Decreases in activity were caused by the agent removing pests.

Pest activity is forecasted using the exponential smoothing method from the R forecasting library (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models, n.d.). An additive trend component captures the increasing population over time. A multiplicative seasonal component captures the

daily cycle in activity. In the time series literature this is also referred to as the Holt-Winters seasonal-multiplicative method and is shown below in equations 4.1.1-4. The variable of interest, y, at some point in time h steps into the future, y_{t+h}, is dependent on the level l_t, the trend b_t, and the seasonal component s_t. The terms α, β, and γ are weights found via maximum liklihood estimation based on available historical data, and m is the number of steps in a seasonal cycle. See chapter 2 for more details on time series forecasting approaches. The full parameters for this model are provided in Appendix A.1

$$\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m} \tag{4.1.1}$$

$$l_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4.1.2}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{4.1.3}$$

$$s_t = \gamma \frac{y_t}{l_{t-1} - b_{t-1}} + (1-\gamma)s_{t-m} \tag{4.1.4}$$

As section 4.2.5 will discuss shortly, some scenarios will involve missing data. When the missing data is on the end of the time series, the agent simply forecasts longer: from the last measurement, up to the current time, and further into the future as needed. When the missing data is in the middle of the time series history, the time series is first decomposed into seasonal, trend, and error components. Then a linear interpolation is applied to the deseasonalized data, and the seasonal component is added back.

The final dynamic environmental variable is the presence of an annual weed species. As an annual species, the seeds that germinate this season were spread during the previous year. Rather than any daily cycle, the dynamics of the weeds are be a longer, seasonal one. Here the concern is when weeds begin to sprout, and thus require intervention. The weed dynamics are scheduled so that germination begins roughly mid-

way through the experiment, effectively changing the demands on the agent. Every day evaluates the probability of having some number of weeds spawn based on the model, and then the location and hour the weed spawns are chosen randomly.

For forecasting weed activity, a simple exponentially weighted moving average (EWMA) is applied to each time slot in the seasonal cycle, with slots that are 24 hours long. Equation 4.2 below shows the basic form of the EWMA. The expected weed activity on a given day is therefore based on the moving average for that day in previous seasons. As the experiment only lasted for one simulated season, a true updating forecast was not possible, and in practice this was an estimated average amount of weed activity per day in the season.

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1} \tag{4.2}$$

4.2.2 Charge Behavior

The agent has three behaviors to execute, each associated with one of the key environmental variables: pests, weeds, and sunlight. The charge behavior returns the agent to its charging station, which notionally represents leaving the field to go into direct sunlight. Once the agent docks it begins recharging based on the simulated presence of solar radiation. Table 4.1 lists the component functions that provide activation for this behavior.

Table 4.1: Charge Behavior Activation Components.

Component	Variable	Function	Weight
Motivation Function	Battery Level	$f_1 = 1 - (battery\_level - 0.1)$	$w_1 = 1$

Table 4.1 continued

Releasing Mechanism	Solar Level	$f_2 = \frac{solar\_irradiance}{max\_solar}$	$w_2 = 0.66$
	Constant Factor	$f_3 = 1$	$w_3 = 0.34$
Circadian Function	Solar Level	$D = solar\_irradiance$	$w_c = 0.6$

The charge behavior motivation function is based on the current battery level, scaled so that it activation is maxed when the battery reaches 10%. It's releasing mechanism is based on the current solar irradiance (scaled by the approximate max solar irradiance for a given time step) and a constant factor. This constant factor allows for non-zero activation when there is no sunlight so that the agent may go to charge at night and wait for the sun to come up. As discussed in chapter three, the circadian function is based on a dependency function D that provides an estimate of the utility for executing the behavior during one time step given the environmental state. This dependency function is used together with the forecasted state and expected duration of the behavior to find the best time to activate, and activation at other times is scaled relative to that. See Section 3.4 for the full details. For this behavior, the dependency function is proportional to the solar irradiance. Scaling is not needed, as the circadian function will average constant factors out.

As discussed in chapter three, the motivation function and releasing mechanism are weighted sums of their individual components. If the associated variable of a component is forecasted, then that component becomes a weighted sum of the circadian function's output and the original output. The final activation is the multiplication of the motivation function and releasing mechanism, representing internal and external factors respectively,

and a scaling factor G. For the charge behavior, $G = 1.1$, setting it as higher priority than the pest and weed behaviors. This ensures that when the charge behavior's activation is near its max, it will override other behaviors and keep the agent from becoming stranded without power. Equations 4.3.1-3 show the construction of the final activation function.

$$MOT = w_1 f_1 \tag{4.3.1}$$
$$RM = w_2 f_2 + w_3(w_c CIR + (1 - w_c) f_3) \tag{4.3.2}$$
$$Act = G * MOT * RM = 1.1 * w_1 f_1 * \left(w_2 f_2 + w_3(w_c CIR + (1 - w_c) f_3)\right) \tag{4.3.3}$$

4.2.3 Pest Behavior

There is one pest behavior per quadrant. When activated the agent drives to the closest plant in the relevant quadrant and monitors it for pest activity. Anytime a pest is active at the plant, it is removed. As activity is spread out over the four plants within a quadrant, this means the agent removes 1/4th of pest activity as it monitors. Table 4.2 lists the component functions that provide activation or this behavior.

Table 4.2: Pest Behavior Activation Components.

Component	Variable	Function	Weight
Motivation Function	Time	$f_1 = \frac{hours\_since\_monitor - 48}{120}$	$w_1 = 0.5$
	Constant Factor	$f_2 = 1$	$w_2 = 0.5$
Releasing Mechanism	Distance	$f_3 = \frac{\text{max\_dist} - \text{current\_dist}}{\text{max\_dist}}$	$w_3 = 0.34$
	Pest Level	$f_4 = \frac{pest\_activity}{max\_pest}$	$w_4 = 0.66$
Circadian Function	Pest Level	$D = pest\_activity$	$w_c = 0.6$

The pest behavior's motivation function is based on the time since it last monitored, increasing up to a max at seven days. This drives it to regularly monitor all quadrants, prioritizing the ones visited the least recently. There is also a constant factor, which allows for the agent to continue monitoring a plant even when it has completed the initial action and reset the *hours_since_monitor* variable. This constant factor represents the inherent value of monitoring and plant and removing pests, even if it has been checked recently. The releasing mechanism is based on distance to the nearest plant in the quadrant and the current pest activity level. The dependency function is also based on the pest activity level. The scaling factor G was set to 1. The full activation function is shown in equations 4.4.1-3.

$$MOT = w_1 f_1 + w_2 f_2 \quad (4.4.1)$$

$$RM = w_3 f_3 + w_4(w_c CIR + (1 - w_c) f_4) \quad (4.4.2)$$

$$Act = G * MOT * RM = (w_1 f_1 + w_2 f_2) * \left(w_3 f_3 + w_4(w_c CIR + (1 - w_c) f_4)\right) \quad (4.4.3)$$

The values for the parameters of the pest behavior and the following behaviors were chosen through limited experimental testing in simulation. The process for this is detailed in section 4.2.6. As the round values suggests, they were not overly optimized, just tuned to get the basic desired behavior.

4.2.4 Weed Behavior

The robot has a single weed behavior, which drives around to systematically check each plant for weeds and remove them if found. The activation function for the weed behavior closely matches the pest behavior. Table 4.3 lists the component functions that provide activation or this behavior.

Table 4.3: Weed Behavior Activation Components

Component	Variable	Function	Weight
Motivation Function	Time	$f_1 = \frac{hours\_since\_monitor - 48}{120}$	$w_1 = 0.5$
	Constant Factor	$f_2 = 1$	$w_2 = 0.5$
Releasing Mechanism	Distance	$f_3 = \frac{\text{max\_dist} - \text{current\_dist}}{\text{max\_dist}}$	$w_3 = 0.34$
	Weed Level	$f_4 = \frac{weed\_count}{\text{max }\_weeds}$	$w_4 = 0.66$
Circadian Function	Weed Level	$D = weed\_count$	$w_c = 0.4$

As with the pest behavior, the weed behavior's motivation function is based on the time since it last monitored, increasing to a max at seven days. The constant factor provides non-zero activation from the motivation function when it has checked for weeds so that the agent may continue if there is no other behavior with high activation. Its releasing mechanism is based on distance to the next plant it must check and the current weed level. The dependency function is also based on the weed level. The scaling factor G was set to 1. Once again, equations 4.5.1-3 show the full activation function.

$$MOT = w_1 f_1 + w_2 f_2 \quad (4.5.1)$$

$$RM = w_3 f_3 + w_4(w_c CIR + (1 - w_c) f_4) \quad (4.5.2)$$

$$Act = G * MOT * RM = (w_1 f_1 + w_2 f_2) * \left(w_3 f_3 + w_4(w_c CIR + (1 - w_c) f_4)\right) \quad (4.5.3)$$

4.2.5 Scenarios and Measures

Three scenarios increased the difficulty for the agent in different ways. The first scenario, "Local Sensing", required the agent to directly observe plants to get an updated

measurement of pest and weed levels. This contrasts with the global sensing used in other scenarios, where environmental sensors provided the agent real time information. This can be viewed as a practical case of "slow sensing" for a robotic platform. Most sensors used on robotic platforms perform measurements extremely fast, but the need for a mobile robotic platform to move to an area introduces a delay in collecting the measurement.

For the non-forecasting agent, the most recent measurement taken replaced active sensing of pest/weed levels. In the forecasting condition, both pest and weed levels were forecasted to compensate for the lack of active sensing. In this scenario the relevant measures were the pest and weed levels. The **pest level** is the amount of pest activity per hour averaged over all hours and all four quadrants, while the **weed level** is the number of weeds present per hour averaged over all hours. The hypothesis is **both the pest level and weed level should show a statistically significant decrease in the forecasting condition when compared to the non-forecasting agent**. To test significance for both this scenario and the following scenarios, two-tailed T-tests were applied to simulated data with the standard $P < 0.05$ used as the cutoff for significance.

In the second scenario, "Slow Actuation", global sensing was returned to the agent allowing for immediate sensing of changes in the environment. However, the travel speed of the robot was reduced by a factor of four. This meant it could take several hours to traverse across the entire workspace. In this scenario, the variables of interest are the **pest level** and **charge rate**, as they both have daily cycles and were forecasted. Charge rate is the percent of battery life recharged per hour, averaged over the total time the agent was charging. The hypothesis for scenario two is that **there will be a statistically significant decrease in the average pest level, and increase in the charge rate, in the forecasting**

condition when compared to the non-forecasting agent. The seasonal cycle of weeds is significantly slower, enough that the robot's change in speed will not impact it.

In the final scenario, "Dynamic Weather", the static solar model is replaced using the historical measurements of solar irradiance at a site near Atlanta, GA. This introduces significant amounts of disruptions in solar energy availability. The robot's speed is returned to normal, and the agent maintains global sensing. In this scenario only solar energy is forecasted, making charge rate the only variable of interest. The hypothesis is therefore that **the charge rate for the forecasting agent will have a statistically significant increase compared the non-forecasting agent.** Table 4.4 summarizes the hypotheses for this experiment.

Table 4.4: Summary of experiment hypotheses for each scenario.

Scenario	Hypothesis	Key Metrics
1: Local Sensing	The pest level and weed level should show a statistically significant decrease in the forecasting condition when compared to the non-forecasting agent.	Average pest level Average weed level
2: Slow Actuation	There will be a statistically significant decrease in the pest level, and increase in the charge rate, in the forecasting condition when compared to the non-forecasting agent.	Average pest level Charge rate
3: Dynamic Weather	The charge rate for the forecasting agent will have a statistically significant increase compared to the non-forecasting agent.	Charge rate

4.2.6 Procedure Details

In each scenario a reactive agent is compared to an agent utilizing the artificial circadian system. The only difference between the agents is the inclusion of the circadian function within the releasing mechanism, effectively changing the parameter w_c from zero to non-zero. Scenarios 1 and 2 had hardware trials executed to validate the approach on a physical platform.

There are a number of parameters associated with each behavior. The procedure for selecting these was as follows. A reactive agent was tested in a "null" scenario, without any of the challenges listed in the previous section for the tested scenarios. The agent moved relatively quickly, had global sensing, and weather was static. Parameters were tuned to get the intended qualitative performance, along with reasonable quantitative performance. Then behavior parameters were fixed for all scenarios. The only exception being w_c, which was increased from zero for the forecasting agent condition. Thus, behaviors were never optimized for the inclusion of forecasting. The tested scenarios introduced challenges which degraded agent performance, and the artificial circadian system was applied to test if it could help recover performance, without any separate tuning.

Each scenario was run for 28 simulated days per trial. For the hardware experiment, the environment was sped up 30x compared to real time to reduce a 28 day trial to 22.4 hours. All scenarios had five simulated trials executed. Each perturbed the initial pest population, used separately generated weed germination patterns, and had a different selection of days from historical solar measurements.

4.3 Results

4.3.1 Scenario 1: Local Sensing

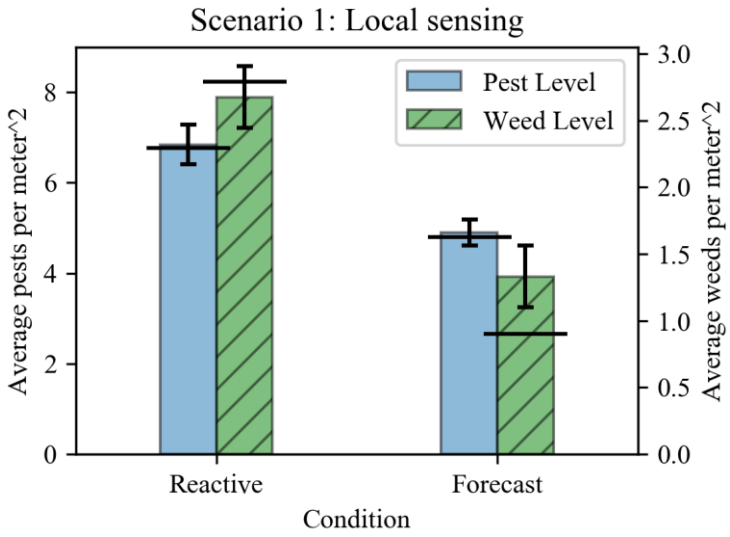

Figure 4.7: Pest and weed levels for scenario 1. The colored bars are the simulated results, with standard error shown. The horizontal lines show the results of the hardware trials.

When comparing the forecasting condition to the non-forecasting condition, pest levels were reduced by 28.3%, and weed levels by 50.2% (P = 0.0059, P = 0.0034 respectively). Figure 4.7 displays the results for both conditions. Hardware results match the pattern shown in simulation. The validates the hypothesis for scenario 1, showing the ACS improved performance.

How did the agent's behavior change with forecasting to improve performance? Figure 4.8 displays histograms of the activation times for the pest behavior. Quadrant 1 is located near the charger, and both agents frequently moved to remove pests from it due to the low cost to do so. The other quadrants required notably more time/energy to travel to, and a clear difference in activation times is shown. The non-forecasting agent executes

sporadically throughout the day. Without any model of the pest dynamics, there's no alignment of the pest behavior with pest activity, as the agent has no way to know when pests are more active without first executing the pest behavior. Instead, the agent generally goes to monitor after it has charged. While pests are still removed during these periods of lower pest activity, it is less efficient.

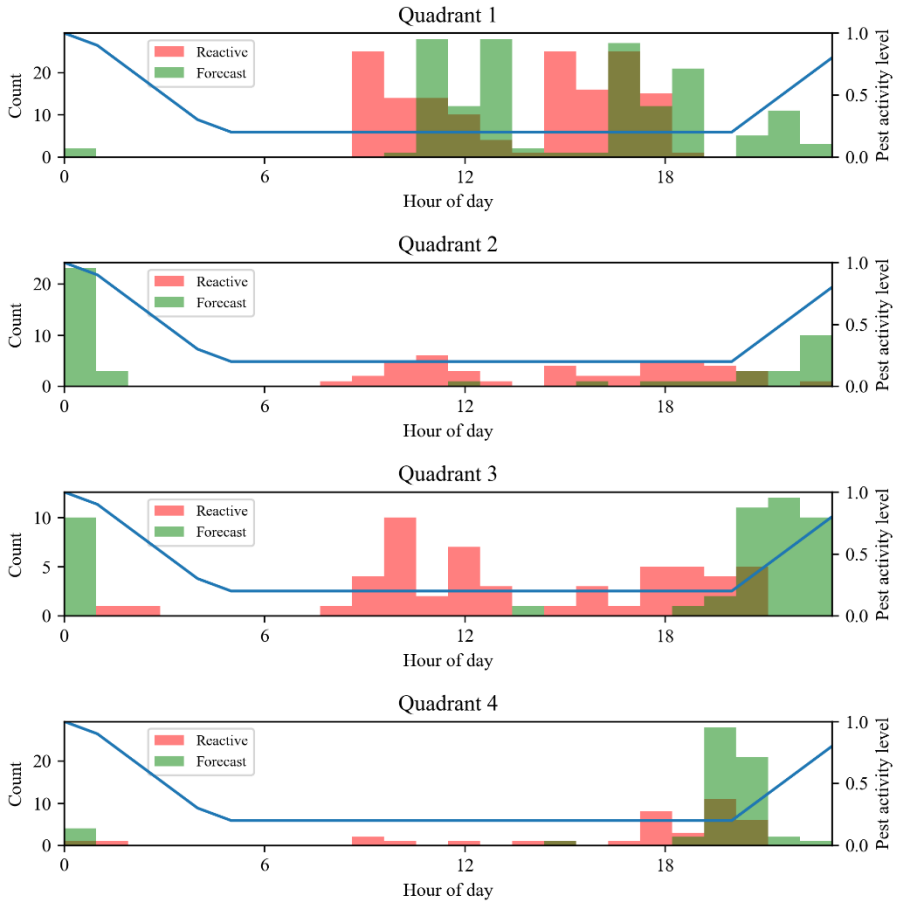

Figure 4.8: Histograms of the pest behavior activation times for each quadrant.

In contrast, the forecasting condition biases the pest behaviors to almost exclusively activate during periods of higher pest activity. The activation also has a noticeable shift forward in time for quadrant 4, which has the longest distance for the agent to travel to or from the charging station, where the agent spends the majority of its time.

Weeds do not have any daily cycle, but the weed behavior is similarly unresponsive without forecasting. With local sensing the agent must estimate weed levels based on its last execution of monitoring, and by chance it may investigate several plants without weeds and incorrectly assume the level of weeds is lower than it is, delaying further work. The forecasting condition balances these noisy measurements with the forecasted germination rates, improving performance.

4.3.2 Scenario 2: Slow actuation

Scenario two's main results are shown in Figure 4.9. In the forecasting condition there is a decrease in average pest levels of 14.3% (statistically significant with P = 0.0017). Though forecasting can't completely overcome the physical limitation of slower actuation imposed on the agent, it helped compensate for it. There is also a small increase in the charge rate of 5.3% (P = 0.0432). However, the hardware results for charging did not match simulation, with the reactive robot charging slightly faster on hardware. The hypothesis for scenario 2 is partially validated. Pest levels were significantly decreased, but charge-rate was not significantly increased.

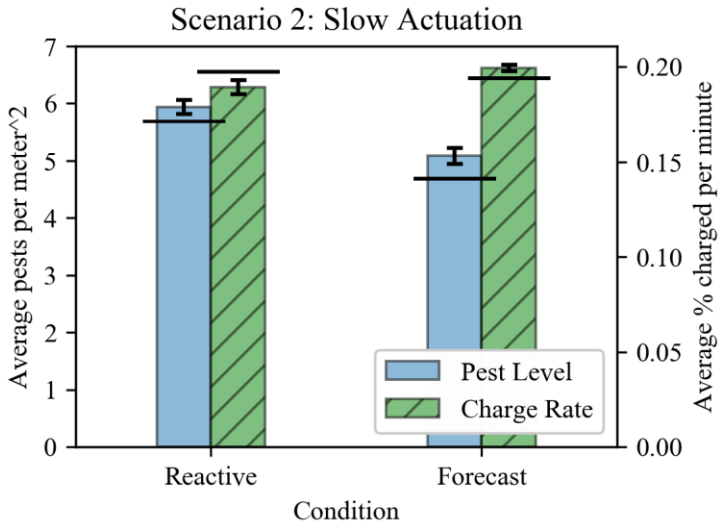

Figure 4.9: Pest and weed levels for scenario 2. The colored bars are the simulated results, with standard error shown. The horizontal lines show the results of the hardware trial.

With regards to pest levels, activation times of the behaviors can be observed in Figure 4.10 to gain insight into the difference. As in the previous scenario, in both conditions the agent would repeatedly go to the first quadrant to clear it off pests due to its short distance from the charging station. For the other quadrants, the activation times of the pest behaviors clearly shift to be earlier for the forecasting agent, in comparison to the reactive one. This is most significant for quadrant 4, the farthest quadrant from the charger, where the agent frequently started from. For quadrant 4 the forecasting agent typically activated the behavior several hours before pest activity increased beyond the minimal level. Under normal conditions this would be detrimental, but the slow agent took several hours to move to a relevant plant. Thus, the forecasting agent successfully generated proactive behavior to deal with its slow actuation.

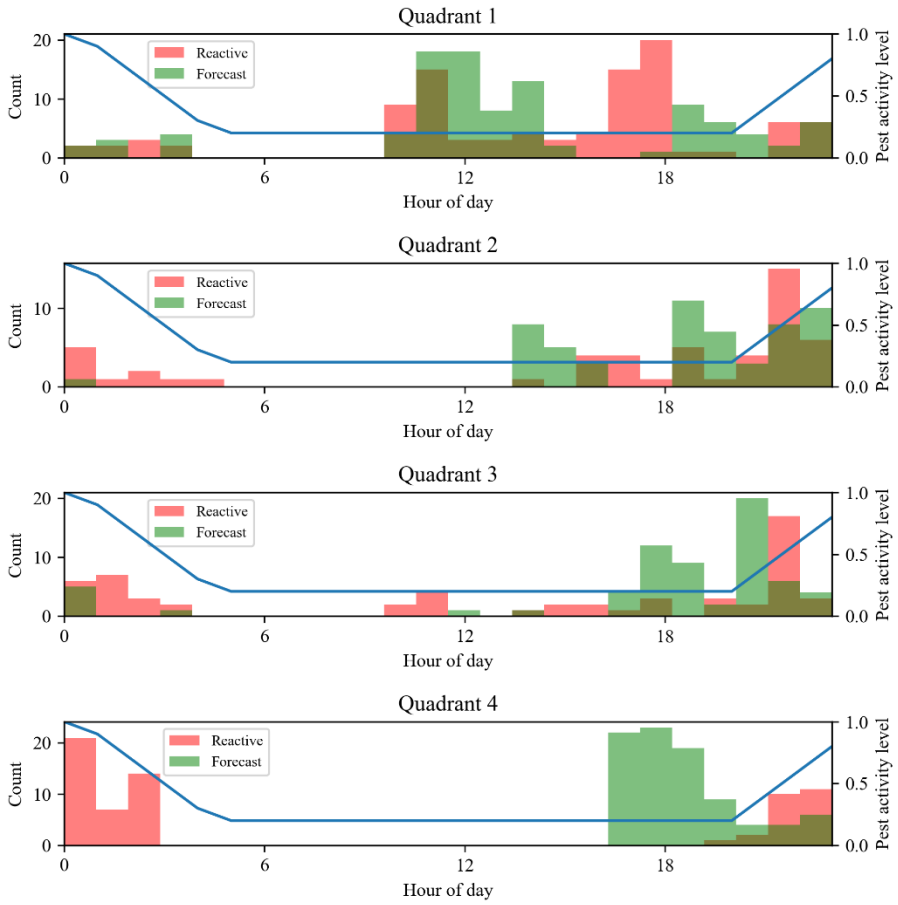

Figure 4.10: Histograms of the pest behavior activation for each quadrant.

The results of the charge behavior are analyzed in the next section along with scenario three.

4.3.3 Scenario 3: Dynamic weather

For scenario three, the only measure was the charge rate. The reactive agent had a charge rate of **0.131** percent/min, and the forecasting agent had a charge rate of **0.128** percent/min. While the reactive agent's average charge rate was slightly higher (~2.3%), statistical tests show that the null hypothesis cannot be rejected: there was no difference between the two conditions (P = 0.508).

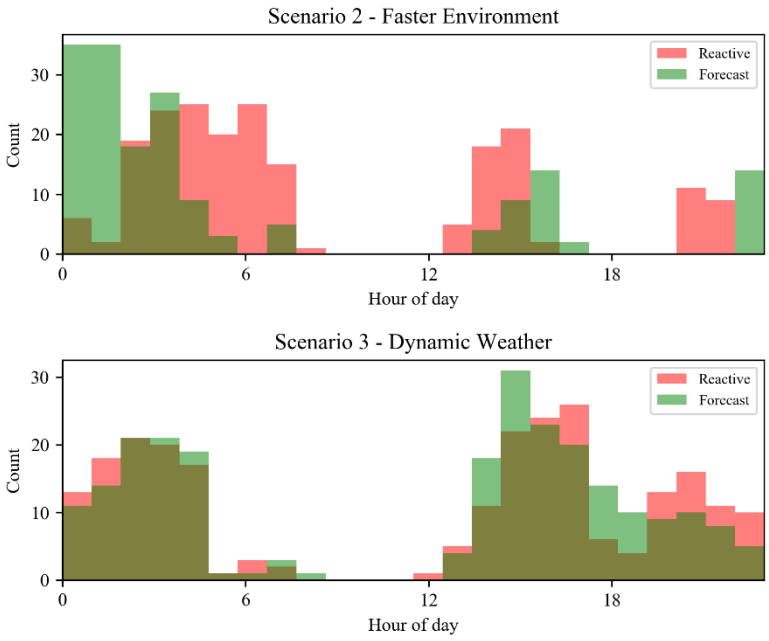

Figure 4.11: Histograms of the charge behavior activation.

The charge behavior activation times for both scenario 2 and 3 are shown in Figure 4.11. These graphs show a strong pattern in both scenarios: begin charging early in the morning, typically before energy is even available (after being depleted at night) and move to charge again in the early afternoon after performing some amount of work. The emergent strategy of expending energy at night limits the ability of the charge behavior to be discriminative about when it chooses to charge.

Look backing at the dynamics shown in Figure 4.4, variation in solar energy happens at two scales. Hour-by-hour there can be significant fluctuation during cloudy weather. Due to the energy and time cost for moving around, the agent is unable to exploit these hourly changes. For reference, the average duration of pest monitoring was around three hours. On a longer scale, some days provide much more energy than others. If the agent could charge enough to run for multiple days, it could choose which days to charge.

Like most solar-powered platforms, daily charging is required, and this was not possible. Due to traits of the agent, it was not able to exploit changing solar energy due to dynamic weather even with forecasted knowledge. This is an important insight discussed further in the following section.

4.4 The Scope of the ACS

One of the goals stated at the beginning of this chapter was to answer the fourth subsidiary research question: "*In what contexts is entraining to a dynamic environment useful for a robotic agent?*". Drawing lessons from these experiments, a set of general rules are proposed. The artificial circadian system provides value when:

- The environment is changing.
- The change impacts an agent's performance.
- The change is forecastable.
- **The environment dynamics are changing on a similar time-scale to the agent's actions.**

The first three points reiterate the discussion in chapter three. If the environment is not changing in a way relevant to the agent, or that change is too random to be predicted, there can clearly by no relevance for forecasting. The last point is the key insight. If the time-scales of the agent's action and environmental change are similar, then the environment may have meaningfully changed once an agent has completed some action, or even during it. This environmental change could significantly impact the utility the agent will receive, and forecasting that change allows for the agent to take it into consideration during action selection, as it did it sections 4.3.1 and 4.3.2.

This is not true for environmental change on time-scales different from the agent. For longer time-scales, an agent will generally have time to sense and react to small changes before they become significant. Forecasting is unnecessary. For change on time-scales shorter than the agent's action, the environmental will change repeatedly before a response could be executed, even with perfect knowledge of the future. Rapid changes will effectively average out. A similar phenomenon is seen in dynamical systems, where time-scale separation can decouple different or components of a system from each other (Sakurama, Verriest, & Egerstedt, 2015).

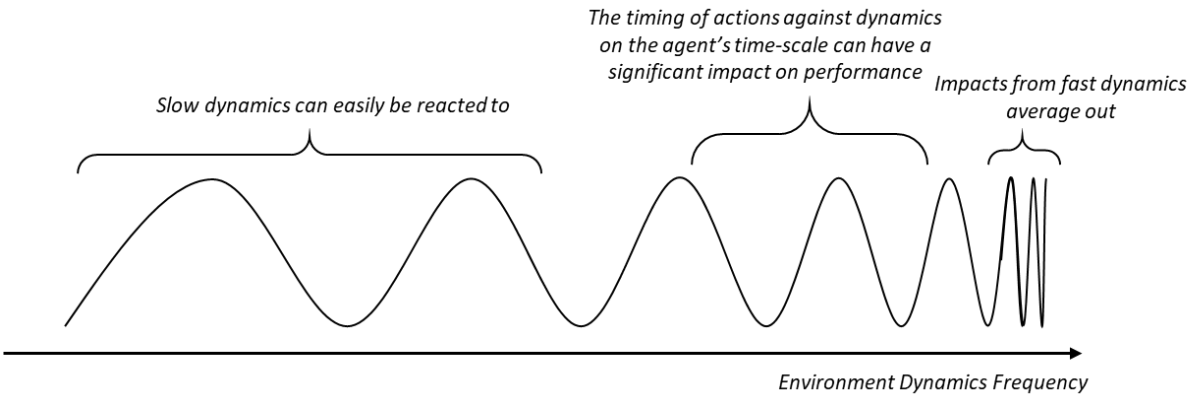

Figure 4.12: An agent will not need, or be able, to entrain behavior to environment dynamics that are much faster or slower than its actions. For dynamics at a similar time-scale, the timing of actions may have a significant impact.

The experimental scenarios detailed in this chapter demonstrated this time-scale rule. In the second scenario, slow actuation, the speed of the agent is slowed down such that travel can take upwards of four hours to traverse the entire workspace. This is similar to pest activity, which is elevated for a roughly eight-hour period every day. This means the agent can miss a large portion of heightened activity if it was not prepared.

In the local sensing scenario, the agent was not particularly slow nor did it consistently monitor too late. Often the agent monitored too early. However, the agent in this experiment was energy constrained, with a limited amount of time it could work. When the agent monitored prematurely, it expended energy and could not fully exploit the periods of high pest activity. As the agent charges every day, this happens on the same time-scale as the pest activity fluctuations. This effect was present in all scenarios, but the local sensing limitation lead to the reactive agent monitoring at worse times more frequently, and thus wasting its limited energy.

The experiment also had dynamics outside of the agents time-scale. Variations in solar energy happened on a scale of hour-to-hour or over multiple days. As mentioned, the agent was unable to exploit either of these. Hour-to-hour is on a time-scale too short for the agent to act, even with perfect knowledge about the future. The energy cost of moving back and forth would overshadow any benefits. The difference in available solar energy over multiple days is on a time-scale too long for the agent, as it had to charge every day.

The rules listed here identify when the features of the environment will cause the timing of an agent's actions to significantly impact its performance. This makes it likely, but does not guarantee, that using the artificial circadian system will provide improvements. There are possible cases where simpler strategies have adequate or even optimal performance. However, if the environment meets these criteria and an autonomous agent is unable to handle the changing environment, then the artificial circadian system (or similar approaches which leverage forecasting) should be considered.

A separate question is whether the ACS is the **right** approach to adapting the agent's behavior, given an environment that meets these criteria. The answer here is similar to what was discussed in chapter 3 on when to use time series models. If you have a complete model of the agent and environment (e.g. a system of dynamical equations, a transfer function, etc), then techniques designed for those models (e.g. control theory, Markov decision processes) may do better. In this work, time series forecasting provides some insight to the future of the environment, but it is not assumed that you have a complete model of how the environment changes and how the agent impacts it. Rather the ACS exploits what it has, forecasts for some variables. If a reasonably accurate model for all relevant state variables is available, approaches that can exploit that have the potential to perform better.

4.5 Conclusions

In this chapter, the artificial circadian system was tested in a notional precision-agricultural task. Multiple scenarios stressed the agent in different ways. Quantitative results demonstrated the ACS was able to improve performance in some cases. Qualitative analysis showed how it entrained agent behavior to the dynamics of the environment, timing the activation of behaviors around the dominant cycles. Drawing from the results, an answer to the fourth subsidiary question was provided. A set of rules is proposed to define the general scope of the ACS based on the environment and how it interacts with the agent.

In the first chapter of this dissertation, it was mentioned that both reactive controllers and set schedules can be used to deal with changing environments. This chapter

has focused on comparing the artificial circadian system to a reactive agent, but what of an agent with a hardcoded schedule designed offline to best exploit a regular environmental cycle? An obvious issue is that such an approach will break down if (or when) the environments dynamics deviate from their normal cycle. In the next chapter, responding to change within the environment dynamics will be covered.

CHAPTER 5. ROBUSTNESS TO CHANGING DYNAMICS

It is inevitable that for any real robotic platform, models of the environment will eventually fail. Simplified models may not capture important effects, truly random or unpredictable events may occur, or the dynamics themselves may change over time. In these situations, forecasting may not only fail to help, but may make an agent's performance worse than a more naïve reactive approach. The work in this chapter addresses these challenges and answers the third subsidiary research question: "*Can such a system be made robust to changes in the dynamics, due to either short-term perturbations or long-term environmental change?"*.

The first section of this chapter details extensions to the ACS to make it robust to deviations of the environment from its model. The second section covers a modified version of the experiment in chapter four. The experiment will include new scenarios where the dynamics undergo both temporary and permanent shifts. Both the original ACS and a new robust version are tested on each scenario. The third section discusses the results of the experiment, including quantitative and qualitative analysis, before the chapter concludes.

5.1 Robustness in the Artificial Circadian System

The goal of robustness in this context is to make the ACS fail-safe with regards to inaccurate forecasts. The ACS can be considered robust if it can leverage forecasting when

it is accurate and ignore forecasting when accuracy degrades. This involves three steps that will be examined in this section:

- Identify when forecasting accuracy has degraded
- Adjust action-selection to not rely on the degraded forecasting
- Attempt recovery of forecasting

5.1.1 Identifying Modelling Error

The most important capability for achieving robustness is the ability to detect when forecast accuracy is degrading. It is impossible to know how accurate a forecast generated at the current moment is in advance. Instead, the accuracy of forecasts in the past are evaluated using measurements taken by the agent after the forecasts were generated. Similar to the forecasting process itself, an assumption is made that forecast accuracy in the past predicts forecast accuracy in the future. If yesterday's forecast for today was accurate, today's forecast for tomorrow may be as well. This assumption is broken whenever the environment's dynamics transition. There will always be a delay between the forecast becoming inaccurate (or accurate again) and the ability of the agent to detect that change. For cases of short-term disruptions, however, the agent can explicitly check this assumption and potentially speed up recovery. Section 5.1.3 below will discuss this in more detail.

For a state variable y_t, the standard error of its forecast $\hat{y}_t$ at some point in time is defined as:

$$e_t = y_t - \hat{y}_t \tag{5.1}$$

Many approaches can be used to evaluate the error over time. Standard metrics such as root means square error (RMSE) or the mean absolute error (MAE) provide absolute measures of error. Percentage errors can be used to achieve a unit-independent measure, which allows for the metric to be applied in different domains more easily. The work in this dissertation uses a metric based on the mean absolute scaled error (MASE) proposed in (Hyndman & Koehler, 2006) and shown below in equations 5.2 and 5.3. The idea is to scale the measured error by the error of the naïve forecast. The naïve forecast always predicts the next time step as the value of the previous time step. Thus, the error of the naïve forecast is always the difference between the values of two consecutive time steps. The MASE looks at the mean of this scaled error over multiple time steps. MASE will be below one whenever the forecasting model is performing better than the naïve approach.

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \tag{5.2}$$

$$MASE = mean(|q_t|) \tag{5.3}$$

Without any model of the environment and how it changes, the best a robotic agent can do is to rely on direct sensor measurements of the relevant environmental variables. Whenever the variable can't be actively measured, the best estimation is the last measured value. Though not typically referred to as a naïve forecast, this assumption that the most recent measurement is the most accurate guess is commonly applied in robotics and related systems when measurements are not always available; e.g., (Wang, Wang, Li, & Wang, 2016) (Krajnik T. , Fentanes, Santos, & Duckett, 2017).

The metric defined in equations 5.2 and 5.3 is not only a general reference for accuracy but provides a comparison between the accuracy of the forecasts with the assumptions a non-forecasting agent makes. This scaling against the naïve forecast means some level of absolute forecast error will be treated as worse in a simple environment with slow and mild changes, compared to the same level of error in an environment experiencing significant swings. Phrased another way, the error metric MASE is not the absolute error, but how much better (or worse) the forecast is compared to not forecasting. Note that a reactive agent does have one advantage compared to the forecasting agent: it can update continuously. In contrast, time series models are generally discrete and must wait for the next time step to update. For a MASE of one, a reactive agent may perform better due to adjustments made between time steps.

There are a few important variations from the original MASE metric in our approach. Rather than using in-sample data to calculate the naïve forecast error (i.e., the naïve forecast error on the training data for the model), the error for the naïve forecast on the real-time data is used. This scales the forecast error by the estimated accuracy of a non-forecasting agent given the same real-time data. In addition, the agent uses the multi-step forecast for action selection, forecasting a full cycle of the dominant seasonal pattern in advance. It is the accuracy of this multi-step forecast that we are concerned with, and which is used when calculating error, rather than the standard one-step forecasts. Given $\hat{y}_t^k$ is the forecast for time step t generated at time step k, the multi-step error of the forecast at time t for with an environmental cycle of period M is:

$$e_t = \frac{1}{M} \sum_{\tau=t-M}^{t} \hat{y}_\tau^{t-M} - y_\tau \tag{5.4}$$

Rather than taking the mean of this error over all time steps, the value of the last time step is used directly to create a continuously updating measurement of error. The final error metric for a time step t will be referred to as the **scaled error** (SE):

$$SE_t = \frac{e_t}{\frac{1}{M}\sum_{\tau=t-M}^{t}|y_\tau - y_{\tau-1}|} \tag{5.5}$$

Figure 5.1 introduces an example that will be used throughout Section 5.1. Figure 5.1(a) shows an example state variable with a strong cyclic pattern repeating every 24 hours. A simple forecasting method is used which forecasts the value of a one-hour time step as the average of the two time steps at the same hour during the previous two days. The error of the forecast in blue, at timesteps 96 to 119, is used when calculating the multi-step forecast accuracy for time step 120, marked with a vertical blue line. Likewise, the forecast in red is used to calculate the SE for the timepoint marked with the vertical red line.

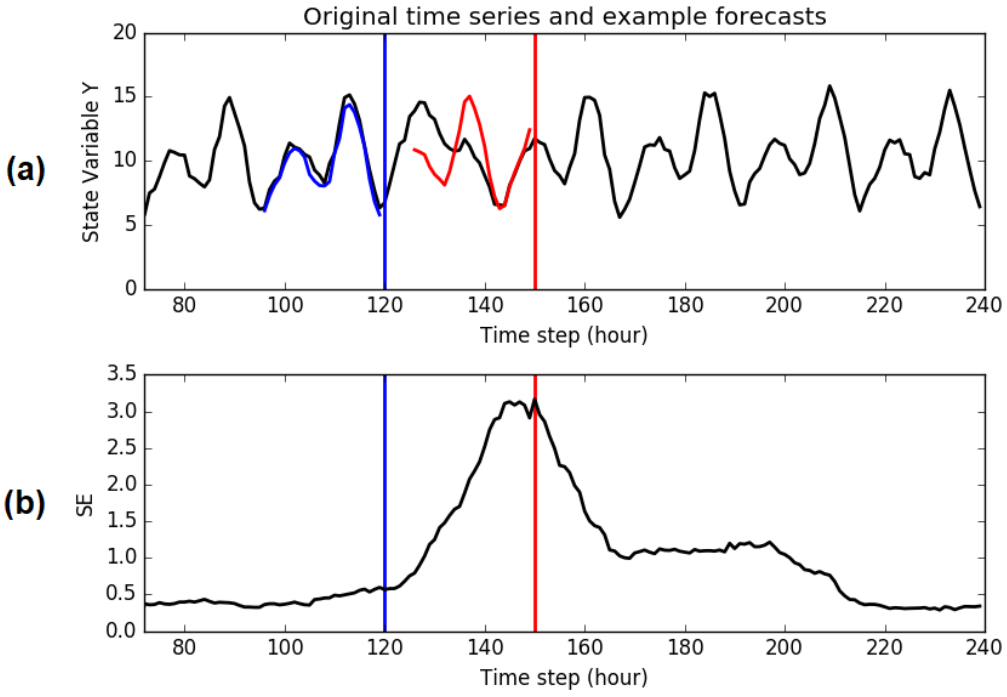

Figure 5.1: (a) shows and example time series with a disruption between hours 125 and 140. (b) shows the forecast error. The blue and red forecasts are used to calculate the SE at the time steps marked by the vertical blue/red lines.

Note the deviation of the state variable from its regular pattern between timesteps 125 and 140. The red forecast, made using data before this environmental transition, shows significantly higher deviation from the true values of Y. Figure 5.1(b) shows how the SE metric defined in this section changes over time. Initially around ~0.5, it spikes dramatically during the period of time the deviation is included in the measurement of accuracy. Afterwards, the SE drops but remains higher than the original value for several cycles due to the deviation biasing the forecasting model. Section 5.1.3 will discuss these phenomena, and potential responses, in more detail. Table 5.1 below provides pseudocode for calculating the scaled error. Note that M is the number of time steps for one period in the cycle.

Table 5.1: Algorithm to calculate the scaled error of a forecast.

Algorithm: Scaled Error
Inputs: • t – Current timestep • state – Vector of state history • forecast – Vector or forecast values generated one cycle (M time steps) in the past
ScaledError(t, state, forecast): **For** i **from** t-M to t: forecast_error += forecast[i] – state[i] naive_error += state[i] – state[i–1] **End For** **Return** forecast_error / naive_error

5.1.2 Adjusting Action Selection

Chapter 3 described how the circadian function is incorporated into action selection. Reviewing briefly, action selection is based on two functions. The motivation function (MOT) and releasing mechanism (RM), which consider internal and external factors respectively. Each are weighted sums of various components. The circadian function comes into play only for the components dependent on the variable being forecasted. It is a function of the forecasted value of an environmental variable, rather than its current measured value. As shown in equation 5.7, the relevant component of the releasing mechanism is replaced with a weighted sum of the circadian function and the original function.

$$RM = \sum_j w_j * f_j(v_j) \tag{5.6}$$

$$f_j(v_j) \;\rightarrow\; w_c CIR(v_j, t) + (1 - w_c) f_j(v_j) \quad 0 \leq w_c \leq 1 \tag{5.7}$$

One option to deal with a degraded forecast is to reduce the length of the forecast being used, e.g. (Droge & Egerstedt, 2011). Forecasts usually become less accurate farther into the future, so this enables the best part of the forecast to still be leveraged. In this work, looking at the forecast over a full cycle is critical. In addition, disruptions in the environment may make forecasts of the near future less accurate than the forecast of later time steps. Instead the weight or influence of the full forecast is reduced. The parameter w_c, which was static in previous work, will be adjusted to respond to changing forecast accuracy. If the SE is equal to one, then the forecast is on average as accurate as the naïve forecast. Around or above values of one, the agent should ignore forecasting entirely and

rely only on sensor measurements. For values below one, the agent transitions to relying more and more on forecasting. The parameter w_c becomes a monotonically decreasing function of SE such that:

$$w_c(0) = L \quad w_c(1) = 0 \quad SE_1 \leq SE_2 \rightarrow w_c(SE_1) \geq w_c(SE_2)$$

L is the maximum value of w_c used in action selection. These constraints provide the general shape of the function $w_c(SE)$, but still leave many possible options: a step function, a linear function, etc. Finding the optimal function would require measuring the performance of the agent with different forecasting accuracies and values for w_c, and is not practical in the contexts this dissertation focuses upon; contexts where collecting samples takes large amounts of time. Instead, a procedure for building a reasonable $w_c(SE)$ function for common use-cases is presented. This procedure was applied in the experiment detailed later in Section 5.3.

A logistic function is chosen to provide a smooth switching behavior; where error values under some level maximize w_c and error values above some level minimize w_c. Given that the max value L is selected as part of tuning action selection, as discussed in chapter 3, this leaves two parameters to select: the midpoint m where the transition is centered, and the growth rate k that determines how fast the transition happens.

$$w_c(SE) = L\left(1 - \frac{1}{1 + e^{-k(SE-m)}}\right) \tag{5.8}$$

For any agent operating in a real environment, there will be some level of forecast error that will exist under normal operation. This will be defined as the **nominal SE**. In the example from Figure 5.1, the average error before any environmental disturbance was just

under 0.5. For simplicity, we will round this up for a nominal SE of 0.5. A reasonable choice for the midpoint m of the logistic function is the midpoint between this nominal SE value and 1. If the system is expected to be more or less sensitive to degrading forecast accuracy, this could be adjusted to be closer or farther from the nominal SE.

$$m = \frac{1 - nominal\_SE}{2} \tag{5.9}$$

The second parameter to set is the growth rate k, which determines the steepness of the function. As discussed above, the constraint to be met is that w_c approaches 0 as SE approaches 1. This can be represented as $w_c(1) = L * \varepsilon$, where ε is some arbitrarily small number. It is important to note that the gain k to achieve this is itself dependent of the midpoint, m. For a higher value of m, closer to 1, the gain must be higher. The gain is therefore a function of a static parameter and m:

$$k = \frac{k_0}{1 - m} \tag{5.10}$$

This can be plugged back into equation 5.8, with $SE = 1$, to solve for the parameter k_0 needed to drive $w_c(1) = L * \varepsilon$ for a certain value of ε:

$$w_c(1) = L * \varepsilon = L * \left(1 - \frac{1}{1 + e^{-\left(\frac{k_0}{1-m}\right)(1-m)}}\right)$$

$$\varepsilon = 1 - \frac{1}{1 + e^{-k_0}}$$

$$1 - \varepsilon = \frac{1}{1 + e^{-k_0}}$$

$$(1 - \varepsilon) + (1 - \varepsilon)e^{-k_0} = 1$$

$$(1 - \varepsilon)e^{-k_0} = \varepsilon$$

$$e^{-k_0} = \frac{\varepsilon}{1 - \varepsilon}$$

$$k_0 = -\ln\left(\frac{\varepsilon}{1 - \varepsilon}\right) \tag{5.11}$$

If the desired value is $\varepsilon < 0.01$, or that $w_c(1)$ will be less than 1% of its maximum value, then apply equation 5.11:

$$k_0 = -\ln\left(\frac{0.01}{1 - 0.01}\right) = 4.595$$

For $\varepsilon = 0.01$ we get $k_0 = 4.595$. Therefore, to hit the requirement of $\varepsilon < 0.01$, we must select

$k_0 > 4.595$. Rounding this up to 4.6, we have a method to set the growth rate k, given any midpoint m, to ensure $w_c(1)$ is within $\varepsilon < 0.01$ of 0:

$$k = \frac{4.6}{1 - m} \tag{5.12}$$

Let us apply this to the example from Figure 5.1. The nominal SE was 0.5. Let us assume for this example the intended max weight of the circadian function for action selection is $L = 0.7$. Applying equations 5.9 and 5.12, we can find the full function $w_c(SE)$:

$$m = \frac{1 - 0.5}{2} = 0.75$$

$$k = \frac{4.6}{1 - 0.75} = 18.4$$

$$w_c(SE) = 0.7\left(1 - \frac{1}{1 + e^{-18.4(SE - 0.75)}}\right)$$

Figure 5.2 below shows the value of $w_c(SE)$ for this example, demonstrating that the desired switching behavior is achieved. When the scaled error spikes, $w_c(SE)$ falls to zero, switching off the impact of forecasting and the circadian component in action selection. Figure 5.3 shows the output of $w_c(SE)$ on this example for a range of midpoint values, with the gain k for each calculated by equation 5.12. The constraints provided ensure they all cut off activation when forecasting is clearly inaccurate. However, the sensitivity to mild increases in forecasting error, and therefore the weight of forecasts in action selection under those conditions, vary. For some sets of values, the output of $w_c(SE)$ is lowered but non-zero for a significant amount of time. Others keep the output maximized up until $w_c(SE)$ goes to zero.

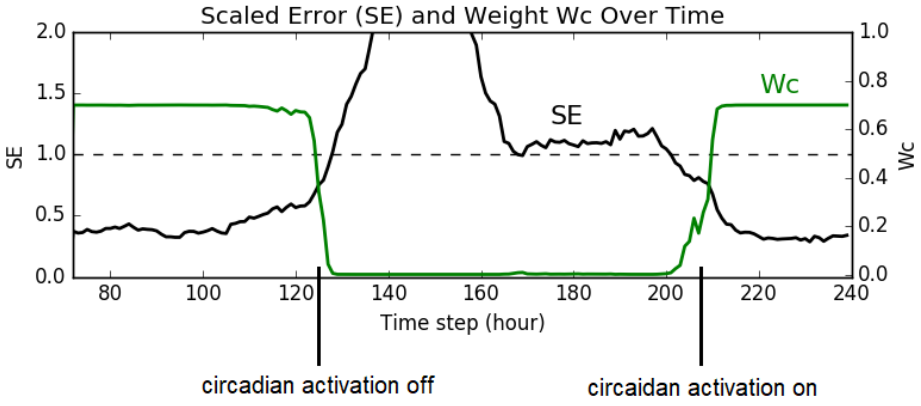

Figure 5.2: The error from the example in Figure 5.1 with circadian weight Wc plotted.

Around time step 125, Wc falls to zero, turning off the usage of the circadian component in action selection. This is reversed around time step 205, when error falls and Wc increases quickly back to its max level.

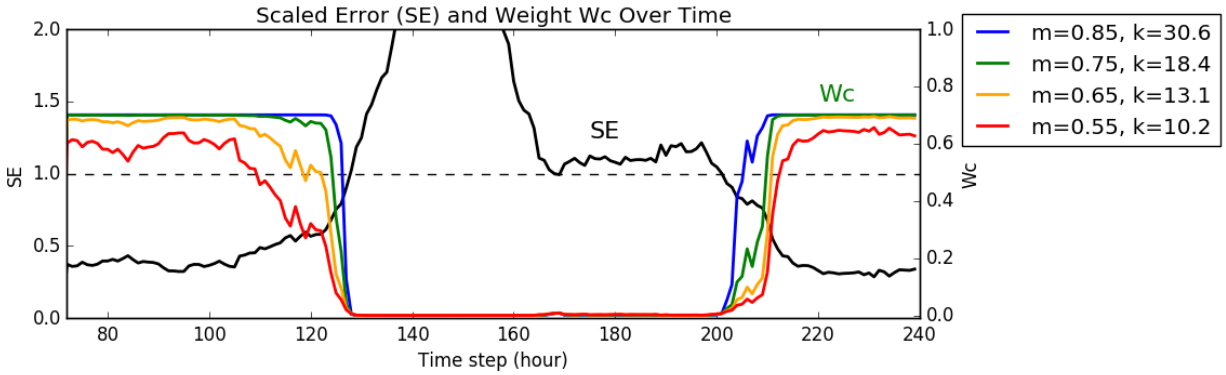

Figure 5.3: The function Wc plotted for several additional values of the midpoint (m) and growth rate (k) parameters. Lower midpoint (m) and gain (k) values means it is more likely for the circadian weight Wc to be reduced without turning off entirely.

5.1.3 Recovery

The approach described so far in this chapter allows the ACS to identify when its model no longer accurately captures the environment dynamics and stops poor quality forecasts from degrading action-selection. The basic recovery option available to an agent is to simply wait until forecasting error returns to normal, if it does. In some cases that may be the best the agent can do. For other cases there are opportunities for the agent to recover faster, if recognized.

Let us define "disruptions" as relatively short-duration changes to the environment, lasting two or less periods of the dominant cycle. Examples of this in the real world include large events in a city disrupting traffic for a day, or serious storms making pests take shelter. While the disruption may end, forecasting based on past data will be less accurate due to

the disrupted history. The number of time steps this effect persists will depend on the model, but it is common for models to leverage data from one to three cycles back.

This creates two distinct phases of error for the forecasting model. Figure 5.4 below visualizes them, using the same simplified scenario introduce earlier for Figure 5.1. Focusing on the normal forecast error in the bottom graph, the initial phase is when data from the disruption event are directly used when calculating the forecast accuracy. This region is marked by the red background region. The second phase, marked by the green background region, is the error introduced by the disrupted data's impact on the forecasting model. Though the error is lower than during and directly after the disruption, it is still significant. Figures 5.2 and 5.3 also show this phenomenon, and how the circadian weight $w_c(SE)$ stays at zero during this second phase.

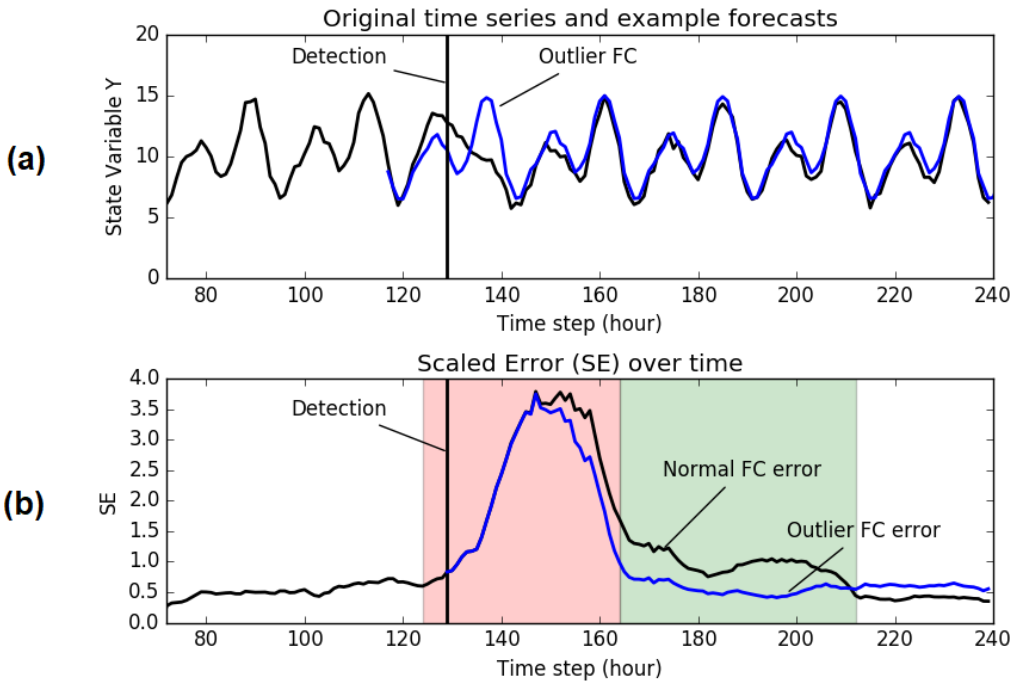

Figure 5.4: A hypothetical disruption event, where the dynamics on day three return to the regular pattern seen before the disruption in day two. The forecast from the current time is

influenced by the disruption, and may have more error then a forecast generated with data before the disruption.

Is there some way the agent can avoid having a temporary disruption reduce forecast accuracy, and therefore performance, for multiple cycles into the future? The agent could ignore recent measurements, which include data from during the disruption, and instead forecast from before the disruption to the current time. Forecasting from farther back in time will increase error and uncertainty, but may provide a better forecast after a disruption. An **outlier forecast** is therefore defined as a forecast based on data from before the environment had changed. An example outlier forecast is shown in Figure 5.4(a) in blue. The example state variable has limited drift, so the forecast is able to project multiple cycles into the future with reasonable accuracy. Given M is the period of the dominant seasonal cycle, S_t is the state history up to timestep t, and $t\_detection$ is the first timestep that the error SE crosses over a threshold determined by the range of SE values under normal operation, then we define the outlier forecast as:

$$t_{outlier} = t_{detection} - \frac{M}{2} \tag{5.13}$$

$$outlier\_FC = forecast\left(\hat{S}_{t_{outlier}}\right) \tag{5.14}$$

How far back does the outlier forecast need to begin? The time step $t_{detection}$ is the time when some change in the environment is detected, but there is no way to know exactly when the change started. Given that the error metric SE uses one cycle of forecasts, the range of options are restricted to between 1 and M time steps, where M is the period of the environmental cycle. Starting the outlier_FC earlier will likely increase error as it has to forecast farther forward in time, but starting too late could leave "bad" data in the forecast,

increasing error even more. Experimental testing suggests a disruption event, one that changes the environment significantly, can generally be detected in ~1/4th of the cycle. To be conservative, in this work the outlier forecast begins from half a cycle before the detection time step. Any environmental change that would take longer to detect, needing the majority of a cycle to drive the error noticeably higher, would be a rather mild change. The forecast from the time step $t_{outlier}$ is generated in the same manner as the normal forecast, but is extended to forecast beyond a single cycle.

The "detection" at this point is not a detection of a disruption specifically, but a detection that the environment changed in some way not captured by the model. Confirming the event is a disruption, as opposed to a more persistent change in the environment, requires confirming that the environment returned to the normal dynamics. The outlier forecast is used to confirm this hypothesis, and if the outlier forecast error falls low enough, it replaces the normal forecast being provided to the action selection mechanism. The threshold used to determine if the outlier forecast is good enough is not the same threshold used to detect the initial disruption. As the scaled error is noisy, two thresholds are used to create a small amount of hysteresis and avoid rapid repeated transitions: **high_thresh** and **low_thresh**. A detection of a potential disruption happens when the normal forecast error goes above high_thresh. To determine that the normal forecast has returned to an acceptable error level, or that the outlier forecast is valid, the scaled error must fall below low_thresh. The exact values of these parameters should be tuned based on the characteristics of the environment; e.g. more noise would suggest more hysteresis is needed and there should be a larger gap between the thresholds. However, a good starting place based on the author's experience is to set high_threshold to be half way

between the nominal_SE and 1 (the same value as the midpoint from equation 5.9) and low_thresh to be 0.05 below that:

$$high\_thresh = \frac{1 - nominal\_SE}{2} \tag{5.15}$$

$$low\_thresh = high\_thresh - 0.05 \tag{5.16}$$

To manage this process which is executed in parallel with the adjustment of the weight $w_c(SE)$, the forecasting component of the artificial circadian system uses a state machine, shown below in Figure 5.5.

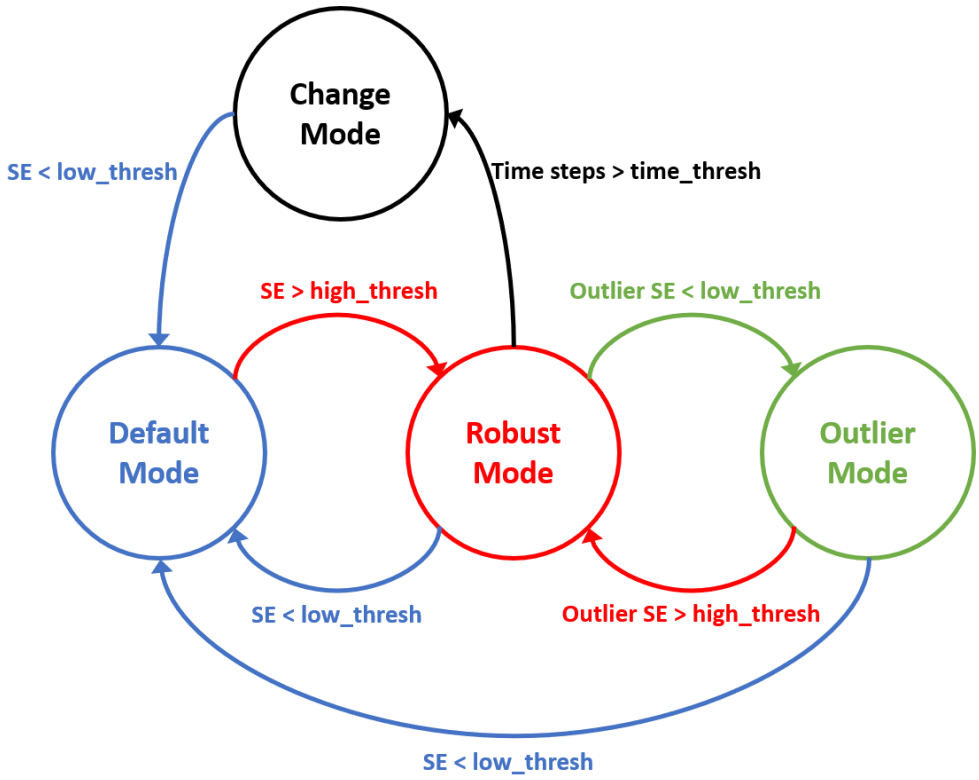

Figure 5.5: State machine for the robust ACS. The robust mode generates and tests the outlier forecast to see if it improves accuracy. The outlier mode provides this outlier forecast to the action selection system. The change mode, if reached, will notify supervisors that a significant change in the environment has happened, and intervention is needed.

In the **default mode**, the forecasts produced and used by action selection are the normal forecasts introduced in chapter 3; a forecast generated based on the most recent measurements. Likewise, the SE used to weight action selection is the error of this normal forecast. If SE crosses the high_thresh, the ACS transitions to the **robust mode**. This mode indicates that change in the environment has been detected, but the ACS continues to use the normal forecast and SE for action selection. In this robust mode the agent also generates the outlier forecast described in this section, and tests the disruption hypothesis using the outlier forecast error. In the case that the event is a disruption, the outlier forecast's SE will still be high during the time the environment is changed (phase 1) but if/when the environment returns to normal (phase 2) the outlier forecast's SE may drop. If outlier forecast error crosses low_thresh, the ACS transitions to **outlier mode**, meaning it has successfully identified an outlier event. In this mode, the outlier forecast and its SE are used for action selection, rather than the normal forecast based on all historical data. The ACS will only stay in outlier mode for a relatively short duration. If the outlier forecast is accurate, then the normal forecast will become accurate again once the model moves past referencing any of the disrupted time steps. Table 5.2 summarizes how each mode of the ACS uses the normal and outlier forecast.

Table 5.2: Forecast Usage in ACS Modes.

Mode	**Normal Forecast**	**Outlier Forecast**
Default	Used for action selection	Not calculated
Robust	Used for action selection	Checked for transition

Table 5.2 continued

Outlier	Checked for transition	Used for action selection
Change	Used for action selection	Not calculated

It's important to note that even with a true disruption event, this approach may not be able to confirm that the event was a disruption and provide a better forecast. Real environment dynamics may be too complicated or noisy to accurately forecast multiple cycles into the future. Particularly if a significant amount of drift exists in the state, forecasts from far back in time may be inaccurate. If so, the outlier forecast error will stay high, and the agent will never transition to outlier mode. In this case, the agent will simply continue to ignore forecasts in action selection (due to high SE) until the normal forecasts become accurate again, and the ACS transitions back to the default mode.

On the other end of the spectrum from disruptions, changes to an environment's dynamics may be permanent. In this case, the goal should be to identify the change. As discussed in chapter 2, model selection for time series forecasting is still done predominantly by hand. Some changes may be captured by re-fitting parameters to new data, but not all. Machine-learning based approaches offer the possibility of fully automatic learning, but need significantly more data to train with compared to a given statistical time series model fitting parameters. In the contexts focused on in this dissertation, where one cycle is often an entire day, collecting enough data could take weeks or months. Specialized approaches for learning in these contexts could be interesting future work, but are not addressed in this dissertation.

Thus, the detection and response to apparent permanent changes is more straightforward than for disruptions. If the ACS persists in the **robust mode** for too long, and the outlier forecast is not accurate, then it assumes the environment has changed permanently and transitions to **change mode**. "Too long" is a set number of time steps given by the **time_thresh**. As disruptions are defined to be less than two environmental cycles (2M), and the robust_start time is shifted back half a cycle from detection (M/2), this work sets $time\_thresh = 2.5M$. The only action taken in change mode is to identify this state so that relevant supervisors can be alerted, and appropriate manual interventions taken. As Table 5.2 notes, even if long-term changes are detected, the normal forecast and error are still provided to action selection, though $w_c(SE)$ will likely remain near zero. Thus action-selection will ignore forecasting, and the agent will act reactively until manual intervention updates the environment model.

Table 5.3 below shows the core loop logic the ACS performs every time step. First forecasts are updated, and their error's calculated. Based on those error values, the ACS checks if it should transition to a different state. Finally, based on what mode it is in, it returns the correct forecast and error which are used by the action selection system.

Table 5.3: Algorithm to monitor forecast accuracy and implement robust methods.

Algorithm: Robust ACS Core Loop

Inputs:

- t – Current timestep
- y_history – Vector of state history

Functions:

- Forecast(history) – Forecast with known model using state vector history
  - See Chapter 3
- ScaledError(t, history, forecast) – Finds the scaled error of a forecast
  - See Table 5.1

```
RobustCoreLoop(t, y_history):
   // Step 1: Get the forecasts and scaled errors
   normal_forecast = Forecast( y_history[1:t-M] )        // y_history up to time step t-M
   normal_SE = ScaledError( t, y_history, normal_forecast )
   If ACS_state == (Robust or Outlier):
      outlier_forecast = Forecast( y_history[1:robust_start] )//y_history up to robust_start
      outlier_SE = ScaledError( t, y_history, outlier_forecast )

   // Step 2: Check and transition states as necessary
   If normal_SE < low_thresh:
      ACS_state = Normal
   Else If ACS_state == Normal And normal_SE > high_thresh:
      ACS_state = Robust
      robust_start = t – M/2
      outlier_forecast = Forecast( y_history[1:robust_start] )
      outlier_SE = ScaledError( t, y_history, outlier_forecast )
   Else If ACS_state == Robust:
      If outlier_SE < low_thresh:
         ACS_state = Outlier
      Else If ( t – robust_start ) > time_thresh:
         ACS_state = Change
         Notify supervisory system/personnel
```

Table 5.3 continued

```
    End If
Else If ACS_state == Outlier And outlier_SE > high_thresh:
    ACS_state = Robust
End If

// Step 3: Return the correct forecast/error for action selection to use
If ACS_state = Outlier:
    Return outlier_forecast, outlier_SE
Else:
    return_forecast = Forecast( y_history[1:t] )
    Return return_forecast, normal_SE
End If
```

5.2 Robust Experiment Design

The experiment design presented in this chapter follows the core design detailed in chapter four. The same notional agricultural testbed is used, and the agent still must balance charging from solar energy and removing pests. The "slow actuation" scenario is used as the baseline to create a need for forecasting. The key new element to this experiment is that the environment *dynamics* will change during a trial. A reactive agent, the basic ACS approach, and the robust ACS detailed in this chapter are all tested against various changes to the dynamics. The main hypothesis is that the robust ACS will be able to identify and respond to the changes in the environment dynamics, providing better overall performance than either a reactive approach or the basic ACS. Specific hypotheses to be evaluated are detailed in section 5.2.3.

5.2.1 Environmental Variables

This experiment more narrowly focuses on pest dynamics and agent performance managing pests. To avoid obscuring the impact of changing pest dynamics, weeds are no longer included. Solar energy is still available according to the daily solar cycle, but the static solar dynamics from the previous experiment is used. Days with significant variation due to weather are removed to make available solar power consistent, and it was not forecasted by the agent.

Pest dynamics are still broken down into quadrants where four plants within a quadrant share a single pest population. The underlying pest population uses a linear growth model. A daily activity pattern is used to generate cyclic activity based on the time of day and the total population. When the environment dynamics change, it is this activity pattern that is modified. The underlying population model stays the same. An active pest is displayed on a plant, and the agent can only remove a pest when it is active at the plant being monitored. The agent can therefore only respond to 25% of activity in a quadrant.

This experiment uses an ARIMA based model for forecasting due to a better ability to include other regressor variables besides the historical data. The robot's actions, i.e., the number of pests removed, is used to adjust forecasts and get better estimates of accuracy. The forecasting process starts by applying a Box-Cox transform, defined below in equation 5.17, to the data. This transformation can normalize the absolute differences within a seasonal cycle, even when the cycle is purely multiplicative. Additive seasonal cycles are generally simpler to model (Hyndman R. , Koehler, Ord, & Snyder, 2008). For the simulated pest data, $\lambda = 0.4$ was found to minimize forecast error.

$$y' = \begin{cases} \ln(y) & if \lambda = 0 \\ \frac{y^{\lambda} - 1}{\lambda} & if \lambda \neq 0 \end{cases} \tag{5.17}$$

Next, the data is differenced, so that change in activity is modeled rather than the absolute level of pest activity. This transformed data is then fit with an ARIMA model using the R forecasting library (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, forecast: Forecasting Functions for Time Series and Linear Models). The final model used is an ARIMA(0,0,2)(2,1,1). The first parenthesis, (0,0,2), indicates that a second order moving average model is applied to the data. The second parenthesis, (2,1,1), is the seasonal terms. It indicates one seasonal difference is applied, and the seasonally differenced data is modeled with a second order autoregressive model, and a first order moving average model. To keep the written model concise, the backshift operator B introduced in chapter 2 will be used. As a brief refresher, the backshift operator provides a convenient way to represent differencing:

$$\begin{gathered} By_t = y_{t-1} \\ y_t - y_{t-1} = y_t - By_t = (1 - B)y_t \end{gathered} \tag{5.18}$$

The regressor variable x_t, which is the number of pests removed in this experiment, is included as a standard linear regression. The error in a normal linear regression is replaced with the timeseries model, F_t. The timeseries model therefore captures all change in the state variable y_t not correlated with the regression variable.

$$y_t = \beta_1 x_t + F_t \tag{5.19}$$

The model of forecasted component F_t is shown below where P_i is the seasonal autoregression parameters, q_i and Q_i are the normal and seasonal moving average parameters respectively, M is the period of the seasonal cycle, and ε_t is the error term. Parameters were fit using the R forecasting library on simulated training data. All parameters are provided in Appendix A.2.

$$(1 - P_1B - P_2B^{2M})(1 - B^M)F_t = (1 + q_1B + q_2B^2)(1 + Q_1B^M)\varepsilon_t \qquad (5.20)$$

The forecast from y_t is a forecast of the transformed, differenced data and must be converted back to 'real' units by reversing the differencing and Box Cox transforms. One may ask why data is differenced before the ARIMA model, when the ARIMA model is capable of differencing data on its own. This has to do with the regressor variable, x_t. Differencing beforehand better captures the impact of the removing pests on the long-term activity, as changes in the differenced activity accumulate over time. Differencing within the ARIMA model means that x_t suppresses activity for the time steps it is high but has limited impact after.

Figure 5.6 shows an example forecasting data from one of the simulated trials. The black line is the measured pest activity. The blue line is the forecast from the current time forward, using the assumption the agent does not remove any pests. The red line is the forecast from 24 hours ago up to now, using the actual robot's actions as the regressor variable. This visualizes the accuracy of recent forecasts by overlaying them onto the real data. The inclusion of the actual regressor data means the forecast shown in red is not the same as the forecast actually made 24 hours ago, but the regressor variable prevents the robot's actions from adding large amounts of error to otherwise accurate forecasts.

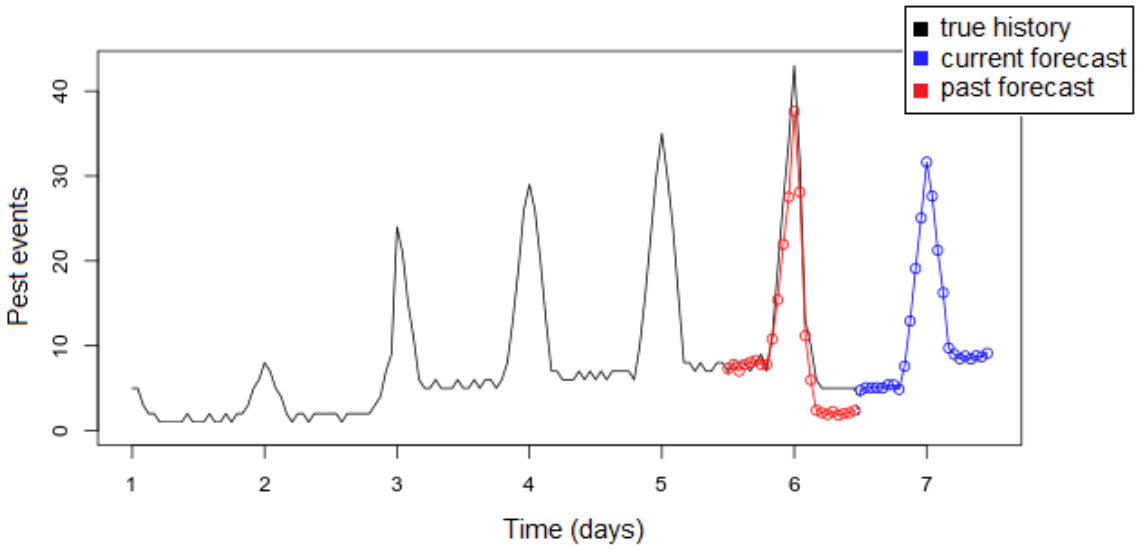

Figure 5.6: The pest activity of one quadrant over several days is shown above in black. The forecast forward from the current time is in blue, while the forecast from 24 hours ago is shown in red. The forecast model is able to do a good job predicting pest activity, even including pest removals by the agent which create the dip in pest activity just after day 6.

5.2.2 Behaviors

The agent has two behaviors to execute: pest monitoring and charging. The charge behavior returns the agent to its charging station, which notionally represents leaving the field to go into direct sunlight. Once docked, the agent begins recharging based on the presence of solar radiation. Table 5.4 lists the component functions that provide activation for this behavior.

Table 5.4: Charge Behavior Activation Components

Component	Variable	Function	Weight
Motivation Function	Battery Level	$f_1 = 1 - (battery\_level - 0.1)$	$w_1 = 1$
Releasing Mechanism	Solar Level	$f_2 = \frac{solar\_irradiance}{max\_solar}$	$w_2 = 0.66$

Table 5.4 continued

Releasing Mechanism	Constant Factor	$f_3 = 1$	$w_3 = 0.34$

Activation for the charge behavior is the same as in chapter 4. The motivation function is based on the current battery level and scaled to reach max activation when the battery is at 10%. The releasing mechanism is based on the current solar irradiance and a constant factor. The constant factor allows for non-zero activation even without sunlight. This means the agent can move to charge at night before the sun has come up.

There is one pest behavior per quadrant which drives the agent to the closest plant in the relevant quadrant and monitors it for pest activity. Anytime a pest is active at the plant during this monitoring, the agent removes it. Activity is spread out over the four plants within a quadrant, thus the agent removes pests from 1/4th of the activity. Table 5.5 lists the component functions for this behavior.

Table 5.5: Pest Behavior Activation Components.

Component	**Variable**	**Function**	**Weight**
Motivation Function	Time	$f_1 = \frac{hours\_since - 48}{120}$	$w_1 = 0.5$
	Constant Factor	$f_2 = 1$	$w_2 = 0.5$
Releasing Mechanism	Distance	$f_3 = \frac{\text{max\_dist} - \text{current\_dist}}{\text{max\_dist}}$	$w_3 = 0.34$
	Pest Level	$f_4 = \frac{pest\_activity}{max\_pest}$	$w_4 = 0.66$
Circadian Function	Pest Level	$D = pest\_act$	$\text{w}_\text{c}(SE) = 0.6\left(1 - \frac{1}{1 + e^{-30(SE - 0.85)}}\right)$

The pest behavior's motivation function is based on the hours since it last activated, with a maximum at seven days. This means the agent regularly monitors all quadrants, and prioritizes the ones not visited recently. There is also a constant factor, which allows for the agent to continue monitoring a plant even when it was completed the initial action and reset the *hours_since* variable. This constant factor represents the inherent value of monitoring and plant and removing pests, even if it has been checked recently. The releasing mechanism is based on distance to the nearest plant in the quadrant and the current pest activity level. The dependency function used with the circadian function is also based on the pest activity level. The weight of the circadian function is changed to a function of the scaled error. Where previously it was hardcoded to 0.6, that is now the maximum value. Based on experimental testing, the limit for nominal SE was identified as 0.7. Applying equations 5.9 and 5.12, the midpoint is set to 0.85 and the growth rate to 30. This means when SE the crosses above 0.7, the weight w_c rapidly decreases.

5.2.3 Scenarios, Measures, and Hypotheses

Three scenarios change the environment dynamics in different ways. The first scenario is **shifted disruption**. For a period of 36 hours, the activity pattern is shifted and scaled such that periods of high pest activity are out-of-sync with the original pattern. After the disruption, activity returns to the original pattern. The second is **random disruption**, which also changes the dynamics for 36 hours, but provides random amount of activity at each hour by shuffling the activation per hour. Shuffling is used to keep the total activation the same. The final scenario is **permanent change**. The activity pattern is adjusted to follow a shorter cycle (18 hours instead of 24) which persists for the rest of the trial. Figures 5.7-5.9 below show examples of these dynamic changes from simulated trials.

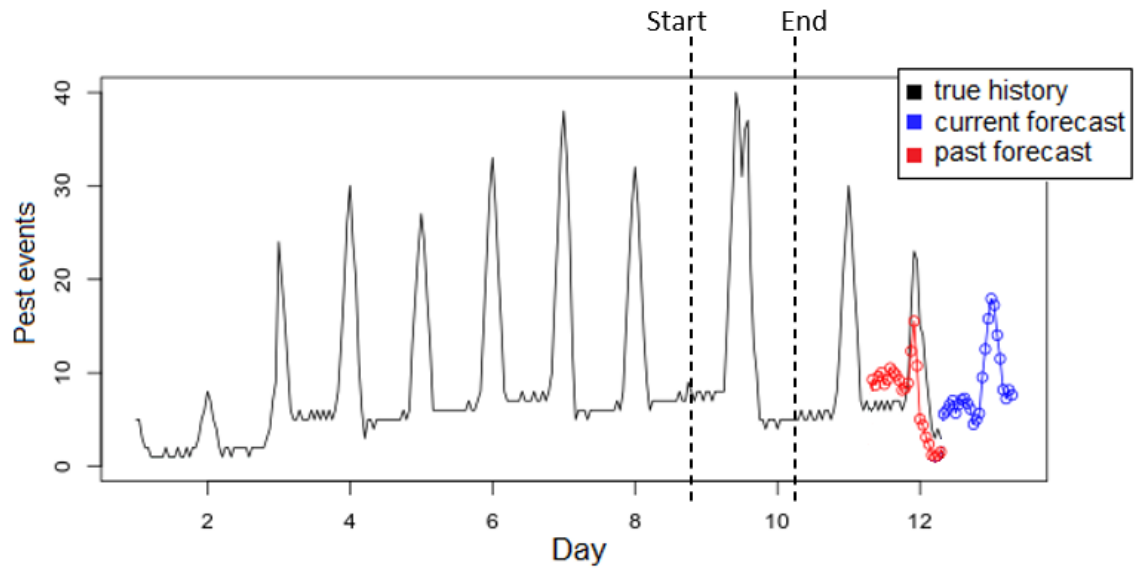

Figure 5.7: Example pest history (black) from the shifted disruption scenario. The 'start' and 'end' lines mark the period the dynamics are changed. The peak of activity around day 9 is wider and offset from the regular cycle. Though the dynamics returned to normal, the forecasts (shown in red/blue) are still degraded.

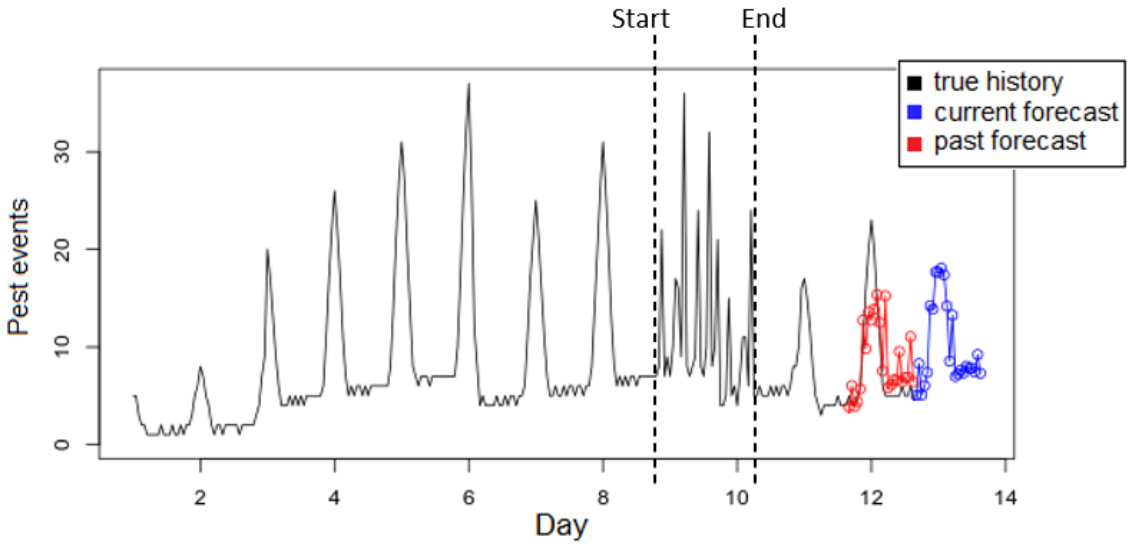

Figure 5.8: Example pest history (black) from the random disruption scenario. The 'start' and 'end' lines mark the period the dynamics are changed. Pest activity during this disruption is randomly shuffled hour-by-hour. The accuracy of forecasts (red/blue) generated afterwards are again degraded.

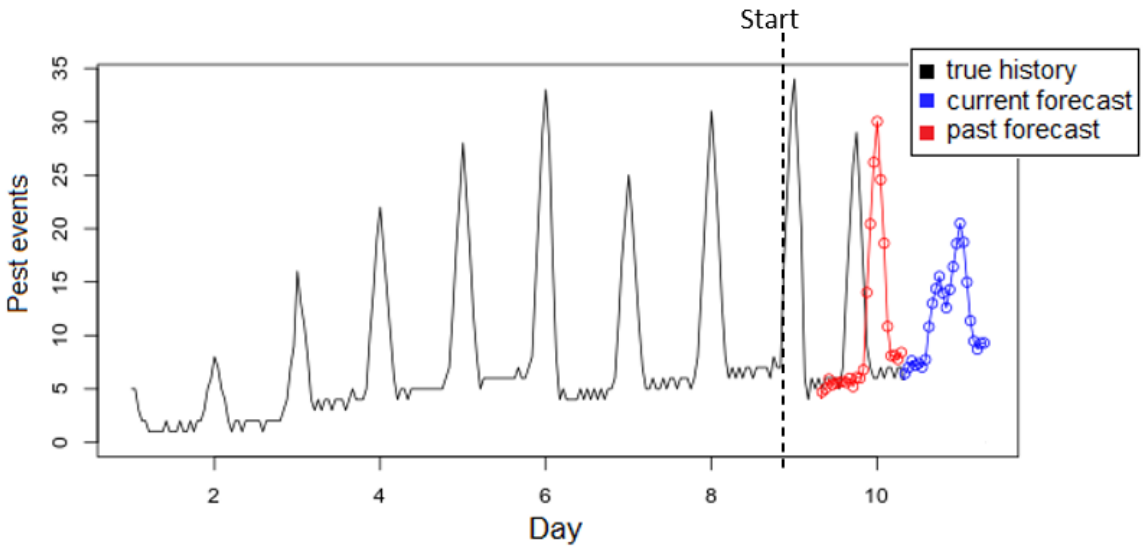

Figure 5.9: Example pest history (black) from the permanent change scenario, where the cycle's duration is reduced. The 'start' line marks where the dynamics change. The past forecast (red) was generated with data just after the change to the environment and highlights how the new pest activity is offset from the predicted activity.

The key quantitative metric for all experimental conditions is the **pest-level post-change**: the pest population averaged over all hours and all four quadrants, for time steps after the environment dynamics changed. Several hypotheses are presented. In each case, better performance means a lower pest-level:

H1. In the disruption scenarios, the basic-ACS will outperform the reactive agent.

The basic-ACS is expected to have degraded performance during environment disruptions. However, disruptions are short, and based on previous experiments (see chapter 4) the basic-ACS should perform better than a reactive agent during the rest of the experiment. Thus, it is expected that over the entire duration, the basic-ACS will still perform better than a reactive agent.

H2. In the disruption scenarios, the robust-ACS will outperform both the basic-ACS and the reactive agent.

The robust-ACS is expected to adapt to these environmental changes, as designed, and thus reap the same benefits as the basic-ACS when forecasting is accurate without having performance degrade as much when forecasting is poor. If the robust-ACS can accurately identify and switch when forecasting error increases, it will perform better than either the basic-ACS or a reactive agent.

H3. In the permanent change scenario, the reactive agent will outperform the basic-ACS.

As in the disruption scenarios, the inaccurate forecasts provided by the basic-ACS after the environment changes are expected to create worse behavior then a purely reactive approach. In contrast to the disruption scenarios, the permanence means the basic-ACS will have degraded performance for a significant length of time, causing its net performance to be worse than a reactive agent.

H4. In the permanent change scenario, the robust-ACS will outperform both the basic-ACS and reactive agent.

Just like for hypothesis H2, it is expected that the robust-ACS will perform as well as the basic-ACS before the change occurs, and perform as well as the reactive agent after, ensuring it will have the best overall performance.

To test the significance of pest-level differences and validate these hypotheses, the one-way ANOVA and post-hoc testing using Tukey's HSD are applied with the standard $P < 0.05$ used as the cutoff for significance.

5.2.4 Procedure Details

Each scenario was run for 14 days per trial in simulation. All scenarios had five simulated trials executed. Each trial perturbed the initial pest population. Changes to environment happened 138 hours, or 5.75 days into the experiment. This was to allow several days for the agent/environment to stabilize before any change, and then provide a week of data after the change. In the disruption scenarios the dynamics returned to normal 174 hours (7.25 days) into the trial.

5.3 Results

5.3.1 Scenario 1: Shifted Disruption

Table 5.6: Shifted Disrupted Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Robust ACS condition saw a statistically significant decrease in the pest population compared to the No ACS (reactive) and Basic ACS conditions.

	No ACS	**Basic ACS**	**Robust ACS**
Mean Population Post-Change	5.598	5.355	4.878
Standard Dev.	0.352	0.199	0.153

The main results of all three conditions in the shifted disruption scenario is shown above in Table 5.6. A one-way ANOVA indicated the differences between the groups were significant (P=0.002). Post-hoc testing with Tukey's HSD indicated the robust-ACS performed significantly different from the no-ACS and basic-ACS conditions (P=0.002 and P=0.027 respectively). In contrast, the difference between the no-ACS and basic-ACS conditions was not significant enough to reject the null hypothesis (P=0.31).

For this scenario we can accept the hypothesis H2; that the robust-ACS would improve performance. Hypothesis H1, that the basic-ACS would improve performance

over the reactive-agent, is rejected. The key factor is how the environment dynamics were changed. In the shifted disruption scenario, the dynamics were not changed arbitrarily but intentionally made to be off-set with the normal dynamics. This could be considered close to the worse-case scenario for the basic-ACS, and some trials showed drastic degradation in performance after the change.

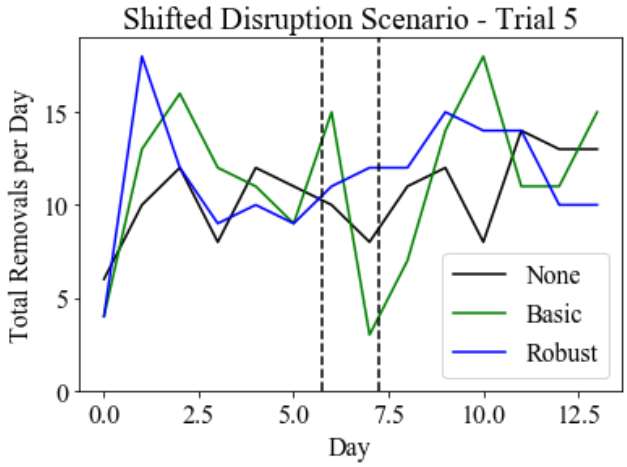

Figure 5.10: Pest removals per day for one trial. The vertical lines mark when the environment was shifted. In this trial, the basic ACS showed a sharp decline in daily removals during the disruption, due to its forecast being out-of-sync with the shifted environment. This did not happen for the no-ACS or robust-ACS conditions.

Figure 5.10 above shows pest removals per day for one trial from the basic-ACS condition, and compares it to the other conditions for the same trial. After the dynamics change, the number of pests removed in the basic-ACS condition falls off as the agent's activity becomes entirely out-of-sync with the environment. Though not every trial saw such a large drop, the average impact was significant. The poor performance during and after the shifted-dynamics cancelled out much of the benefit from forecasting during other portions of the trial.

The robust-ACS was able to reliably identify forecasting has failed, and sometimes was able to successfully generate a disruption-forecast. All quadrants in all trials entered the robust-mode during the shifted dynamics, and 55% entered outlier-mode correctly after the dynamics returned to normal. Figure 5.11 below shows an example of a trial where the outlier forecast provided better predictions for over 24 hours before the normal forecast error reduced. Figure 5.12 shows an example where the outlier mode failed to produce a good forecast. In this particular case, the pest population of the quadrant was low; less than half of the overall average population during the shifted dynamics. The overall low activity meant the agent identified the disruption late and had insufficient data to forecast several days forward with.

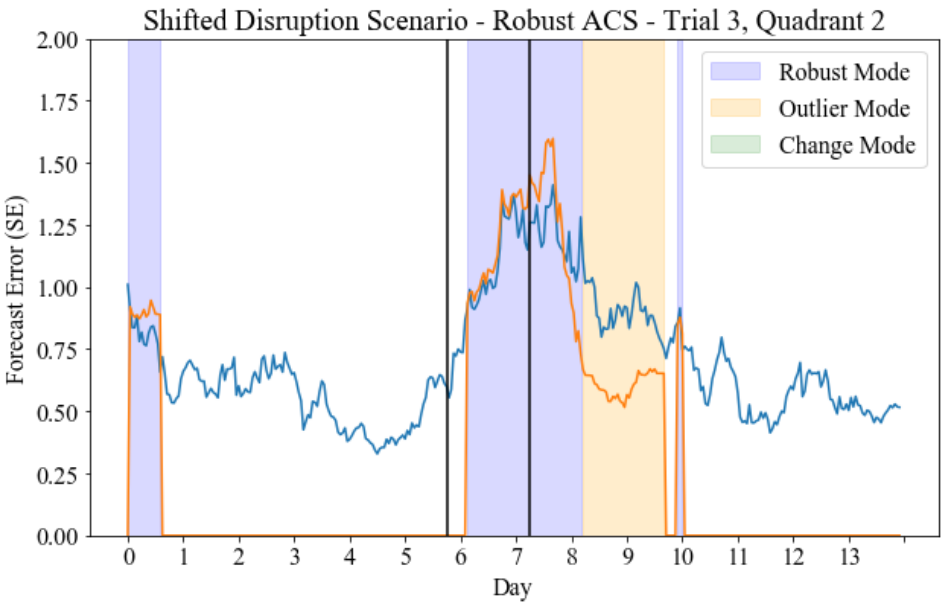

Figure 5.11: The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the shifted dynamics. In this trial/quadrant, the outlier was quickly identified (see the blue region beginning in day 6, identifying that the robust mode started). An outlier-forecast successfully generated accurate predictions and transitioned the robust ACS to outlier mode (shown as the yellow region) until the normal forecast error returned to adequate levels.

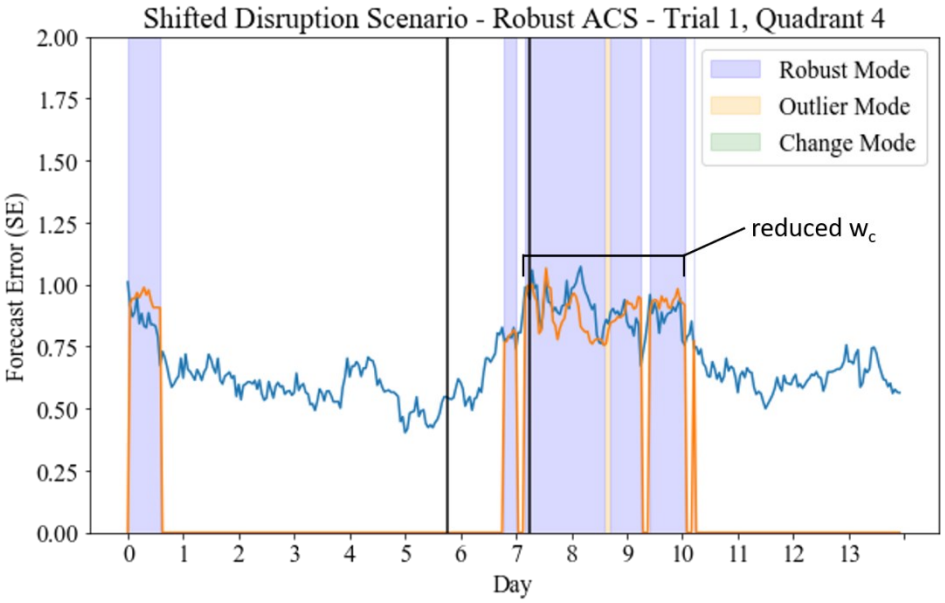

Figure 5.12: The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the shifted dynamics. In this trial/quadrant, the outlier was identified late, and a good outlier-forecast was never generated. However, the robust-ACS still reduced the weight of the circadian component after the disruption. Through most of day 7 to day 10, Wc is significantly reduced due to the increased forecast error. This mitigates the worst outcomes from inaccurate forecasts.

5.3.2 Scenario 2: Random Disruption

Table 5.7: Random Disruption Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Basic-ACS and Robust-ACS conditions both had significantly lower pest population from the No ACS condition, and were not significantly different from each other.

	No-ACS	**Basic-ACS**	**Robust-ACS**
Mean Population Post-Change	5.918	4.79	4.67
Standard Dev.	0.338	0.256	0.143

Table 5.7 displays the main results for the random-disruption scenario. Once again, a one-way ANOVA showed significant differences between the groups ($P<0.001$). Post-hoc testing showed that the no-ACS condition was significantly different from both forecasting

conditions (P=0.001 for both), and the two forecasting conditions were not significantly difference (P=0.71).

The first two hypotheses (H1 and H2) presented in section 5.2.4 apply to the random disruption scenario as well as the shifted disruption scenario. For the random disruption scenario, the first hypothesis H1 is validated. The basic-ACS improved performance over the no-ACS condition. The second hypothesis H2, that the robust-ACS would outperform the basic-ACS, is rejected. The randomized activity meant that no intelligent timing of activity could be generated by the agent. The accuracy of forecasting, or the usage of forecasting at all, had no apparent impact during the disruption.

The randomized disruption interacted with the scaled error (SE) in a counterintuitive way. An important detail to remember is that the SE is not a measure of how accurate forecasts are, but how accurate they are relative to the naïve forecast. Randomization is the worst-case for a naïve forecast. This means that SE stayed low during the random disruption, due to the naïve forecast error increasing just as much. Figure 5.13 below shows an example of how the SE was unaffected during the disruption, but spikes after as the dynamics return to normal due to the forecast still using the randomized data.

The key difference when compared to scenario 1 is that while the random dynamics degraded forecast accuracy, it did not do so in a way that biased activation towards any specific time. The basic-ACS did not see a significant degradation of performance, either during the random disturbance or after due to forecasting. Therefore, the robust-ACS had little to compensate for.

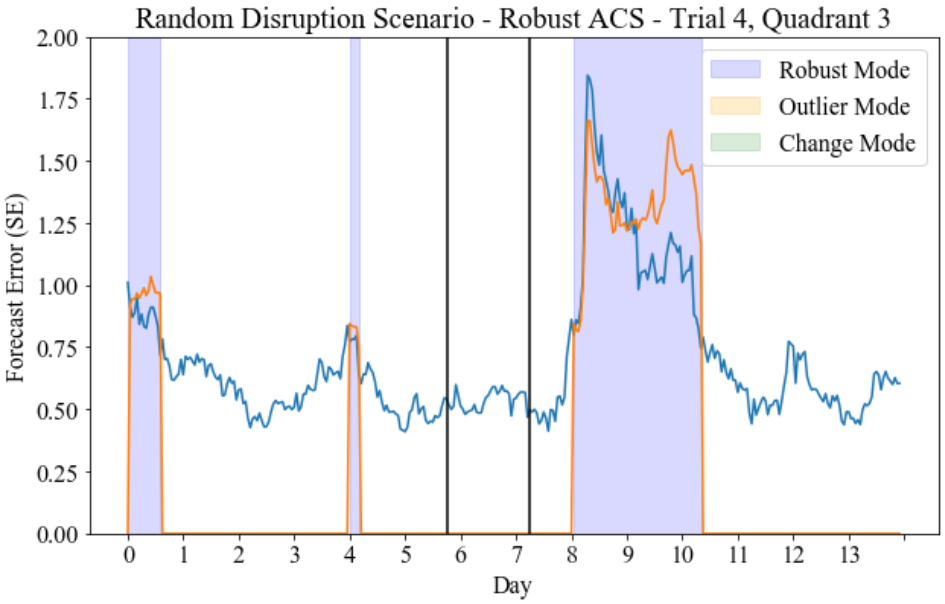

Figure 5.13: The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The two black vertical lines indicate the beginning and end of the random dynamics. Due to the naïve forecast error increasing at the same time, the SE does not increase during the random disruption. Instead, SE spikes after the dynamics return to normal, but when data from the disruption is still impacting forecasting.

5.3.3 Scenario 3: Permanent Change

Table 5.8: Permanent Change Results. The mean and standard deviation of the pest population per quadrant after the environment changed is shown. The Robust ACS condition had a pest population significantly lower compared to the No ACS (reactive) and Basic ACS conditions.

	No-ACS	**Basic-ACS**	**Robust-ACS**
Mean Population Post-Change	6.162	5.932	5.387
Standard Dev.	0.441	0.19	0.214

The third and final scenario, permanent-change, has the main results presented above in Table 5.8. One-way ANOVA showed the conditions were significantly different (P=0.005), and post-hoc testing showed the robust-ACS condition was significantly different from both the no-ACS condition (P=0.004) and the basic-ACS condition

(P=0.0365). There was no significant difference between the no-ACS and basic-ACS conditions (P=0.477).

The fourth hypothesis H4 was that the robust-ACS would outperform both the no-ACS and basic-ACS conditions in this scenario, and this hypothesis is validated. In addition, the robust-ACS successfully identified that the environment changed in 95% of quadrants over all trials. It was able to take advantage of forecasting early on, ignore forecasting after the environment dynamics changed, and alert any supervisory system or personnel to the change. Figure 5.14 below shows how the change was detected in one example quadrant.

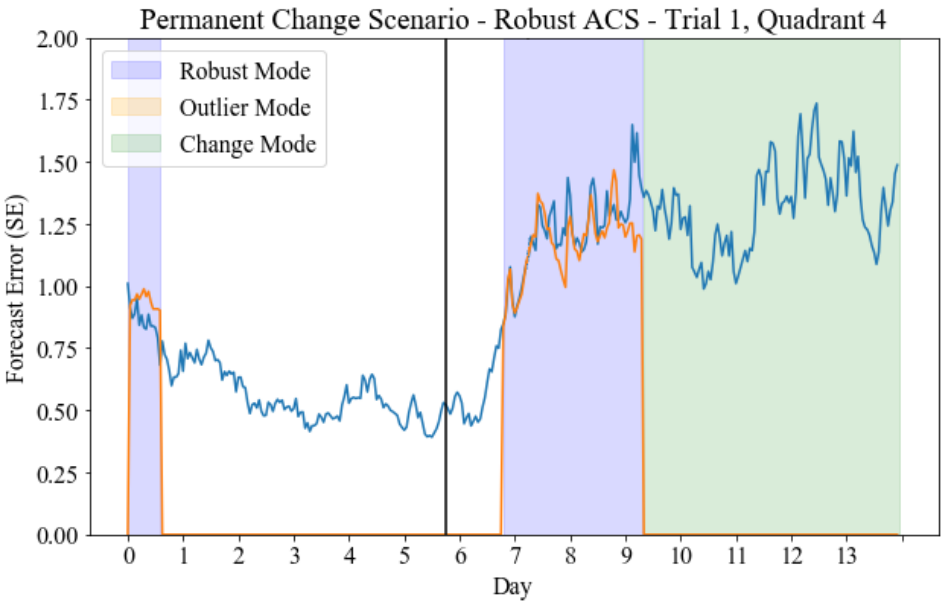

Figure 5.14: The normal SE (blue line) and outlier SE (orange line) plotted over the duration of the experiment. The black vertical line indicates when the dynamics change. In this trial, the robust-ACS quickly enters robust mode after the dynamics change and error increases. Once enough time has passed, it transitions to change mode, representing that it has detected a long-term change to the environment.

The third hypothesis H3, that the no-ACS condition would perform better than the reactive-ACS condition, was rejected in this experiment. However, analyzing the trends in

more detail suggest that with more time, the basic-ACS's overall performance would have degraded further. Figure 5.15 below shows the total pest populations of all quadrants averaged over all trials in the permanent-change scenario. Initially, the normal and robust ACS conditions maintain similar performance, both decreasing the total pest population. After the environment dynamics changed, both forecasting conditions see increasing pest populations. The robust-ACS approaches the population of the no-ACS condition which aligns with expectations, as it will not be using forecasting. The basic-ACS shows a pest population growing faster and overshooting the no-ACS condition.

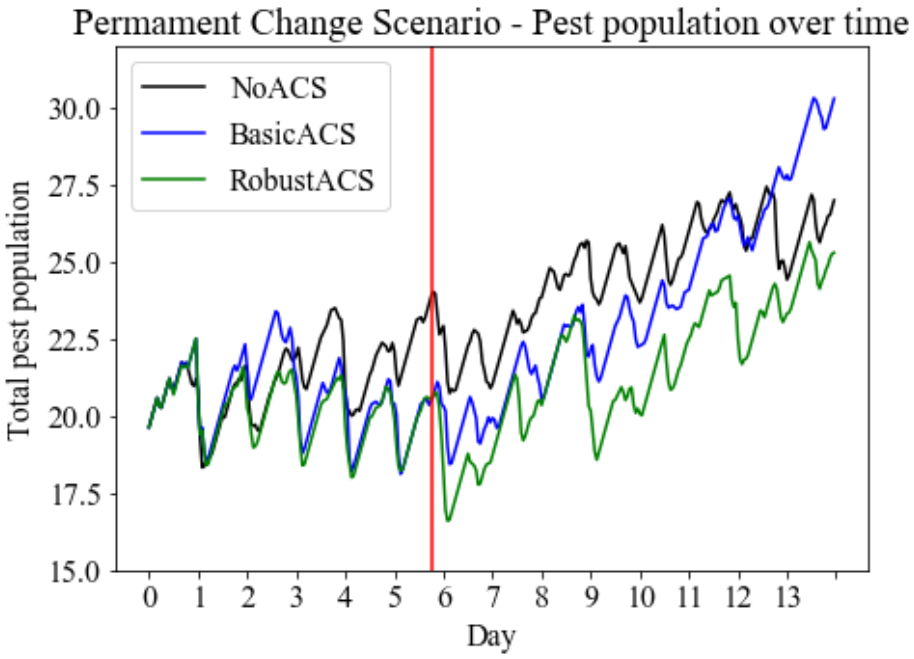

Figure 5.15: The total pest population averaged over all trials. Before the environment change, marked by the vertical red line, both the basic and robust-ACS conditions diverge from the no-ACS condition. After the change, the robust-ACS looks to be approaching the no-ACS condition, while the basic-ACS sharply rises above both.

5.4 Conclusions

This chapter developed and validated extensions to the artificial circadian system to make it robust to changes in the environment dynamics, addressing the third subsidiary research

question. This included a method for monitoring forecast accuracy in real time, how action-selection should be adjusted based on changing forecast accuracy, and two hypotheses the ACS can test to help recover faster.

Several experimental hypotheses were developed to compare the basic-ACS, the new robust-ACS, and a purely reactive agent. Hypotheses H1 and H3 focused on the performance of the basic-ACS, predicting it would perform better than the reactive agent in the disruption scenarios, but worse in the permanent change scenario. Results showed that in the shifted-disruption and permanent change scenarios, the basic-ACS had equivalent performance to the reactive agent. In the random-disruption scenario, it performed significantly better. The key takeaway is that the basic-ACS may see significant degradation in performance when the environment dynamics change, performing no better than a reactive agent not forecasting at all. How sensitive the basic-ACS is to these changes depends on how exactly the dynamics change.

Hypotheses H2 and H4 were concerned with the performance of the robust-ACS and they predicted it would outperform both the basic-ACS and reactive agent in all scenarios. The robust-ACS did perform significantly better than the reactive agent in all scenarios, and better then the basic-ACS in all scenarios except for random-disruption (where it performed equivalently). In summary, the robust-ACS was able to perform as good or better than either the basic-ACS or a reactive agent in every scenario. The key capability of the robust-ACS was reliable detection of when the environment no longer matched its internal forecasting model. This allowed action-selection to be adjusted to compensate. Additionally, in some cases it was able to identify that a short-term disruption

had occurred and generate accurate forecasts by ignoring the data from during the disruption.

The robust-ACS allows for forecasting to be used much more opportunistically. If the robotic platform has an adequate non-forecasting-based strategy it can fall back on, then the ACS can be applied. Even if the forecasting models of the environment, or the environment itself, are not reliable. Beyond the original goal of robustness, this expands the range of potential applications.

CHAPTER 6. CONCLUSION

The world is in constant change. The solar and seasonal cycles of earth drive both weather and the behavior of living organisms. These patterns are engraved into human activity, and thus still impact most man-made environments. As robotic platforms inevitably become persistent features in many of these environments, they will experience and be impacted by these dynamics. Circadian clocks are a well-known natural adaption to dynamic environments, and their prevalence in a large variety of organisms suggests that persistent robots may also benefit from some circadian-like system to adapt to a changing environment. This leads to the core research question of this dissertation: *How can the long-term dynamics of an environment be modeled, predicted, and exploited by a slow and persistent autonomous robot?*

This research question was broken down into four subsidiary questions. In this chapter they will each be reexamined and results relevant to them will be summarized. After, the contributions made in the pursuit of these questions will be presented. Finally, future extensions will be discussed and the work concluded.

6.1 Research Questions Revisited

1. How can the long-term dynamics of an agent's environment be characterized or modeled?

The core capability a robotic agent requires to adapt to a changing environment is an understanding of how the environment is changing. Based on the inspiration of this work, a natural first idea would be biomimicry of circadian systems. This dissertation summarized how natural circadian systems work, and some of their limitations (Section 2.1). Circadian systems can be closely approximated by a phase-only model which cannot capture non-cyclic dynamics or represent changes in peak amplitudes of a variable. The fundamental problem a robotic platform faces is time series forecasting: predicting the future of an environment (specifically a state variable) based on its history. While traditionally applied offline to static data sets, statistical time series forecasting offers more generality in what it can model and can provide detailed forecasts (Section 2.2). The application of time series forecasting online on an autonomous robot was presented (Section 3.3). The effectiveness of the leveraging these types of models was validated for a simplified precision agricultural task through the results of several experiments in which an autonomous agent relied on these forecasts to make better decisions (Sections 4.3, 5.3).

2. How can individual behaviors, and the set of active behaviors, adapt to changes in the environment using the knowledge of environmental dynamics?

An activation-based approach was used for action selection. In this framework, the question is narrowed to how forecasts should impact activation of a behavior, and thus change both a behavior's individual timing and the patterns of activity of the agent overall. Chapter 3 introduced a key assumption, that the dynamics would be strongly cyclic with a known period, as this is the case for many environments on earth. In this context, there is a natural time-horizon for an agent to use when looking forward: the period of the dominant cycle. The agent only needs to know the best timing of a behavior within the cycle. This

dissertation presented methods and algorithms for computing the utility of executing a behavior at any point in the next cycle, and weighting the current activation based on the relative utility (Section 3.4). This was dubbed the circadian function, and provided activation for a behavior based on how its timing over the dominant cycle. This circadian function was used through the experiments in chapters 4 and 5. Detailed graphs of the activation times demonstrate how the circadian function helped synchronize activity with the environment (Figures 4.8 and 4.10).

3. Can such a system be made robust to changes in the dynamics, due to either short-term perturbations or long-term environmental change?

All models are approximations, and environments will continue to change over time, making the original approximation less accurate. Leveraging inaccurate forecasts could create worse performance then simple reactive responses. Chapter 5 dived deeply into how the artificial circadian system could be made robust to the dynamics themselves changing. Given a measure of forecast accuracy (Section 5.1.1), a robotic agent can remove the influence of bad forecasts in action selection (Section 5.1.2). This allows an agent to leverage the models when they are helpful, and fall back to a reactive approach when they are not. Forecast models will fail when the environment itself changes, and these changes can be either permanent or temporary. Due to the reliance on historical data for prediction, a short-term disruption to the environment can potentially degrade forecast accuracy for days during which the environment has returned to normal. Methods were presented to exclude outlier data from the forecast model and leverage this modified forecast if it improved accuracy (Section 5.1.3). Experimental results showed the artificial circadian system (ACS) could reliably identify when simulated pest dynamics changed and

forecasting degraded in the agricultural testbed (Section 5.3). Sometimes outlier data was also successfully removed to generated modified forecasts and help forecasting recover faster, but this was more difficult and not consistently successful.

4. In what contexts is entraining to a dynamic environment useful for a robotic agent?

Not all cyclic behavior in nature is generated by circadian rhythms (Section 2.1). A reactive response to a cyclic environment will create cyclic behavior. Likewise, just because an environment is changing does not necessarily mean a robotic agent needs to model and forecast the changes to effectively deal with it. The experiments in chapter 4 were designed to explore different contexts to better understand the scope of this work. A set of conditions under which the ACS will have value were developed (Section 4.4). Some of the conditions are straightforward: the environment must be changing in a way that impacts the agent and is predictable. A key insight is that the dynamics must change at a similar time-scale to the agent's ability to act. Dynamics that change slowly (relative to the agent) can be handled reactively, and do not need a more complex forecasting approach. On the other end of the spectrum, an agent will be unable to exploit dynamics changing faster than it can act, even if the forecasts of the dynamics are perfect. Slow robots, along with being more capable of persisting for long durations, are more likely to act on a similar time-scale to their environment's dynamics.

6.2 Contributions

- **Algorithms to apply time series forecasting in an autonomous robotic agent.**

While some examples of autonomous robots modeling the dynamics of their environment exist (e.g. Krajnik T., Fentanes, Santos, & Duckett, 2017), the author is not aware of any which leverage time series models. In this work, standard models for time series forecasting were applied online in an iterative manner, continuously providing an autonomous agent an updated forecast of key environmental variables (Section 3.3). Chapters 4 and 5 presented experiments applying this approach with different underlying time series models under different conditions, including with missing data (Section 4.3.1), validating it as an approach an autonomous robot can effectively use.

- **Algorithms and methods to estimate the utility of executing a behavior given a forecasted environment state and integrate this into action selection.**

Novel methods were presented that consider the estimated delay to execute a behavior, the estimated duration, the estimated environment state (given by forecasts), and the estimated performance based on the environment state. These are used together to estimate the utility of activate a behavior at any time (Section 3.4.1). The relative utility of executing the behavior now, compared to the potential utility for waiting, is passed to the action selection system as one component of a behavior's activation. (Section 3.4.3).

- **Methods for an agent to monitor forecast accuracy and adjust its weight in action selection.**

Based on work from (Hyndman & Koehler, 2006), an error metric that provides a real-time measure of forecasting accuracy was presented (Section 5.1.1). This is a unitless measure that compares a forecast to the implicit assumption a non-forecasting agent makes, that the measured state will stay the same. Novel methods for tuning the weight of forecasting

within action selection using this metric were described (Section 5.1.2). These methods let an acceptable level of error be specified and have the agent smoothly transition to ignoring forecasts as accuracy degrades.

- **Algorithms to identify disruptions in the environment and ignore outlier data to recover forecasting faster.**

This dissertation demonstrated novel algorithms to enable an agent to estimate when the environment has been disrupted and exclude this data to create a new modified forecast (Section 5.1.3). This new forecast is leveraged only if it improves accuracy, in which case it replaces the default forecast. This approach was able to recover accuracy significantly faster than waiting for the original forecasting model to correct (Section 5.3.1).

- **Implementation and demonstration on a physical and simulated testbed representing a notional precision agricultural task.**

An agricultural testbed was developed with artificial plants an agent could monitor and interact with (Section 4.1). The agent could also autonomously dock and charge, representing solar charging and permitting long-duration experiments without human intervention. The testbed enabled a variety of dynamics to be tested, including cycles with different lengths, dynamics from simulated models and real data, and variables that could be sensed either globally or locally (Section 4.2). For the robust experiments, these dynamics were also changed midway (Section 5.2). This testbed was applied throughout the work to validate the ideas presented, demonstrating both when the methods worked and when they didn't (Sections 4.3 and 5.3).

6.3 Future work

There are two main directions for future work. The first steps away from robotics and focuses on time series modeling and forecasting. As time series analysis is traditionally done offline on fixed data-sets, a large majority of research and effort have been put towards discrete models and methods. However, robots function in a continuous world. Research has been done on continuous time series models, most commonly continuous ARMA models (Brockwell P. J., 2001). These could potentially be adapted for use online instead of the more common discrete approaches. From an alternative perspective, nonparametric models have been applied to cluster data over cycles in time to achieve continuous modeling of events (Krajník , et al., 2019). Whatever the underlying model is, a continuous implementation of the artificial circadian system could add the ability to reason about the expected time until some event, rather than only estimating the number of events over a period of time. It would also be capable of responding immediately to new significant data, such as a sudden disruption, rather than having to wait until the next time step.

The second area of future work focuses on the underlying action selection method. This dissertation began with the idea of letting a robotic platform persist in a natural environment over long periods of time, becoming part of the ecological system. This led to goal of entraining its behavior to the dynamics, but it also guided the assumptions the work built on. One key assumption was that the robot would not have a complete model of the environment, how it is changing, and how the robot's actions impact it. Rather, forecasting was used to provide useful but limited insight into a complex world. However, some environments and applications do afford more complete models. In such cases,

techniques like Monte Carlo Tree Search or Model Predictive Control may be able to plan an agent's actions, with forecasts of the environment used to predict state changes that are independent of the agent's actions.

6.4 Final words

It is inevitable that the number of robots used in complex and dynamic environments will increase over time. While this dissertation has frequently used agriculture as an example domain, any platform which persists around people will interact with the strongly cyclic pattern of human activity. Adapting to these dynamics will help robotic platforms maximize their performance and minimize inconvenience to the people around them. This dissertation has laid out a complete methodology to model and forecast environment dynamics, incorporate these forecasts into action-selection, and adapt based on changing forecast accuracy.

Using predictions of the future to improve decision making is an intuitive idea. Predicting the future is hard, however. Keeping robots running autonomously for long durations is also hard. Gracefully handling failure is required for any approach to be applicable to the real world. Any method should consider entraining an autonomous agent to its environment as an opportunistic endeavor. Something to be leveraged when possible, but not a fundamental part of action selection. Perhaps unsurprisingly this mirrors natural circadian systems which influence, but do not control, the behavior of animals and other natural organisms.

APPENDIX A. FORECASTING MODEL PARAMETERS

A.1 Chapter 4 Pest Forecasting Model

In the experiment descripted in chapter 4, pest dynamics were modeled with the Holt-Winters seasonal-multiplicative method. This is a type of exponential smoothing method where the trend term is additive, but the seasonal term is multiplicative. The equations of this model are shown below, where $\hat{y}_{t+h|t}$ is the forecasted value of the state y at time step $t+h$, given information up to time step t:

$$\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m} \quad (A.1)$$

$$l_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \quad (A.2)$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \quad (A.3)$$

$$s_t = \gamma \frac{y_t}{l_{t-1} - b_{t-1}} + (1-\gamma)s_{t-m} \quad (A.4)$$

Three smoothing parameters (α, β, γ) determine how fast the level, trend, and seasonal components (l_t, b_t, s_t respectively) update. There is also an initial state for each of the level, trend, and seasonal components. Note the seasonal component is a vector of length 24, with and initial value for each hour of the day.

Table A.1: Smoothing parameters and initial states for Chapter 4 pest model.

$\alpha = 0.75$		$\beta = 0.0001$	$\gamma = 0.0002$
$l_0 =$	0.6357		
$b_0 =$	0.0308		

Table A.1 continued

$s_0 =$	2.405, 1.8461, 1.2896, 0.718, 0.6058, 0.5415 0.5817, 0.5867, 0.5812, 0.5967, 0.5978, 0.5786 0.5852, 0.5823, 0.5578, 0.5758, 0.5702, 0.5737 0.5489, 0.7278, 1.2943, 1.8185, 2.4213, 2.8153

A.2 Chapter 5 Pest Forecasting Model

For the experiment in Chapter 5, pest dynamics used an ARIMA(0,0,2)(2,1,1) model. As discussed, the first parenthesis (0,0,2), indicates that a second order moving average model is applied. The second parenthesis shows the seasonal terms. The values (2,1,1) indicate one seasonal difference is applied, and this seasonally differenced data is modeled with a second order autoregressive model, and a first order moving average model.

The model also uses a regressor term to include the impact of the robot's actions. Equation A.5 shows how the robot's actions, x_t, are included as part of a linear regression which has the error term replaced by the forecast model, F_t. The forecast model is shown in equation A.6, and the full table of parameters below in Table A.2. Note that B is the backtrack operator, discussed in Chapter 2.

$$y_t = \beta_1 x_t + F_t \tag{A.5}$$

$$(1 - P_1 B - P_2 B^{2M})(1 - B^M) F_t = (1 + q_1 B + q_2 B^2)(1 + Q_1 B^M)\varepsilon_t \tag{A.6}$$

Table A.2: Parameters for Chapter 5 pest model.

Regressor terms	$\beta_1 = -0.3376$	---
Autoregressive (AR) terms	---	---
Moving Average (MA) terms	$q_1 = -0.9064$	$q_2 = 0.2778$
Seasonal AR terms	$P_1 = -0.3384$	$P_2 = -0.2337$
Seasonal MA terms	$Q_1 = -0.2401$	---

REFERENCES

Ambrus, R., Ekekrantz, J., Folkesson, J., & Jensfelt, P. (2015). Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario. *In Intelligent Robots and Systems (IROS).*

Arkin, R. C. (1998). *Behavior-based robotcs.* MIT press.

Arkin, R. C., & Egerstedt, M. (2015). Temporal Heterogeneity and the Value of Slowness in Robotic Systems. *In Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on* (pp. 1000-1005). Zhuhai: IEEE.

Arkin, R. C., Fujita, M., Takagi, T., & Hasegawa, R. (2003). An ethological and emotional basis for human–robot interaction. *Robotics and Autonomous Systems, 42*(3), 191-201.

Aschoff, J. (1967). Adaptive cycles: their significance for defining environmental hazards. *International Journal of Biometeorology, 11*(3), 255-278.

Aschoff, J. (1981). A survey on biological rhythms. *Biological Rhythms*, 3-10.

Bacher, P., Madsen, H., & Nielsen, H. A. (2009). Online short-term solar power forecasting. *Solar Energy, 83*(10), 1772-1783.

Bike Arlington. (2020). (Arlington County Commuter Services) Retrieved January 2018, from http://www.bikearlington.com/counter-data/

Bloch, G., Barnes, B. M., Gerkema, M. P., & Helm, B. (2013). Animal activity around the clock with no overt circadian rhythms: patterns, mechanisms and adaptive value. *Proceedings of the Royal Society B: Biological Sciences,*, 280.

Blumberg, B. (1994). Action-selection in hamsterdam: Lessons from ethology. *In Third International Conference on the Simulation of Adaptive Behavior*, (pp. 108-117).

Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2008). *Time series analysis: forecasting and control, 4th edn.* UK: John Wiely & Sons. Inc.

Boyan, J. A., & Littman, M. L. (2001). Exact solutions to time-dependent MDPs. *Advances in Neural Information Processing Systems* (pp. 1026-1032). Vancouver: MIT Press.

Brockwell, P. J. (2001). Continuous-time ARMA processes. *Handbook of statistics, 19*, 249-276.

Brockwell, P. J., & Davis, R. A. (2006). *Introduction to time series and forecasting.* Springer Science & Business Media.

Buchli, B., Sutton, F., Beutel, J., & Thiele, L. (2014). Dynamic power management for long-term energy neutral operation of solar energy harvesting systems. *In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (pp. 31-45).*

Chernova, S., & Arkin, R. C. (2007). From deliberative to routine behaviors: a cognitively inspired action-selection mechanism for routine behavior capture. *Adaptive Behavior, 15*(2), 199-216.

Czeisler, C. A., & Gooley, J. J. (2007). Sleep and circadian rhythms in humans. *Cold Spring Harbor symposia on quantitative biology*, *72.*

Dayoub, F., & Duckett, T. (2008). An adaptive appearance-based map for long-term topological localization of mobile robots. *In International Conference on Intelligent Robots and Systems*, 3364-3369.

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association, 106*(496), 1513-1527.

DeCoursey, P. J. (2004). The behavioral ecology and evolution of biolog- ical timing systems. In J. C. Dunlap, J. Loros, & P. J. DeCoursey, *Chronobiology: Biological Timekeeping* (pp. 27-66). Massachusettes: Sinauer Associates.

Doganis, P., Aggelogiannaki, E., & Sarimveis, H. (2008). A combined model predictive control and time series forecasting framework for production-inventory systems. *International Journal of Production Research, 46*(24), 6841-6853.

Draeger, A., Engell, S., & Ranke, H. (1995). Model predictive control using neural networks. *IEEE Control Systems Magazine, 15*(5), 61-66.

Droge, G., & Egerstedt, M. (2011). Adaptive Time Horizon Optimization in Model Predictive Control. *Proceedings of the 2011 American Control Conference* (pp. 1843-1848). IEEE.

Essen, H. v., & Nijmeijer, H. (2001). Non-linear model predictive control for constrained mobile robots. *European Control Conference* (pp. 1157-1162). IEEE.

Fentanes, J. P., Lacerda, B., Krajnık, T., Hawes, N., & Hanheide, M. (2015). Now or later? Predicting and Maximising Success of Navigation Actions. *In International Conference on Robotics and Automation (ICRA).*

Fleury, F., Allemand, R., Vavre, F., Fouillet, P., & Bouletreau, M. (2000). Adaptive significance of a circadian clock: temporal segregation of activities reduces intrinsic competitive inferiority in Drosophila parasitoids. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 1005-1010.

Franklin Robotics. (2020). *Tertill: A robot that weeds your garden*. (Franklin Robotics) Retrieved January 2018, from http://www.franklinrobotics.com/

Gardner, E. S. (1985). Exponential smoothing: The state of the art. *Journal of forecasting, 4*(1), 1-28.

Gardner, E. S. (2006). Exponential smoothing: The state of the art. *International journal of forecasting, 22*(4), 637-666.

Gardner, M. J., Hubbard, K. E., Hotta, C. T., Dodd, A. N., & Webb, A. A. (2006). How plants tell the time. *Biochemical Journal*, 15-24.

Gu, D., & Hu, H. (2002). Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems, 39*(2), 73-86.

Hahnel, D., Triebel, R., Burgard, W., & Thrun, S. (2003). Map building with mobile robots in dynamic environments. *IEEE International Conference on Robotics and Automation, 2*, 1557-1563.

Hall, D. C., & Norgaard, R. B. (1973). On the timing and application of pesticides. *American Journal of Agricultural Economics, 55*(2), 198-201.

Hardin, P. E., Hall, J. C., & Rosbash, M. (1990). Feedback of the Drosophila period gene product on circadian cycling of its messenger RNA levels. *Nature, 343*(6258), 536-540.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 5-10.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy,. *International journal of forecasting, 22*(4), 679-688.

Hyndman, R., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: the state space approach.* Springer Science & Business Media.

Hyndman, R., O'Hara-Wild, M., Bergmeir, C., Razbash, S., & Wang, E. (2019). *forecast: Forecasting Functions for Time Series and Linear Models*. Retrieved July 2019, from The Comprehensive R Archive Network: https://cran.r-project.org/web/packages/forecast/index.html

Hyndman, R., O'Hara-Wild, M., Bergmeir, C., Razbash, S., & Wang, E. (n.d.). *forecast: Forecasting Functions for Time Series and Linear Models*. Retrieved July 2019, from https://cran.r-project.org/web/packages/forecast/index.html

Johnson, C. H., & Golden, S. S. (1999). Circadian programs in cyanobacteria: adaptiveness and mechanism. *Annual Reviews in Microbiology, 53*(1), 389-409.

Kansal, A., Hsu, J., Zahedi, S., & Srivastava, M. B. (2007). Power Management in Energy Harvesting Sensor. *ACM Transactions on Embedded Computing Systems (TECS), 6*(4).

Kindlmann, P., Arditi, R., & Dixon, A. (2004). A simple aphid population model. *Aphids in a new millennium. INRA Editions*, (pp. 325-330). Paris.

Klančar, G., & Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and autonomous systems,, 55*(6), 460-469.

Krajník , T., Vintr, T., Molina, S., Fentanes, J. P., Cielniak, G., Mozos, O. M., . . . Duckett, T. (2019). Warped hypertime representations for long-term autonomy of mobile robots. *IEEE Robotics and Automation Letters, 4*(4), 3310-3317.

Krajnik, T., Fentanes, J. P., Santos, J. M., & Duckett, T. (2016). Frequency map enhancement: introducing dynamics into static environment models. *ICRA 2016 Workshop on AI for Long-term Autonom.* ICRA 2016 Workshop on AI for Long-term Autonomy.

Krajnik, T., Fentanes, J. P., Santos, J. M., & Duckett, T. (2017). Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. *IEEE Transactions on Robotics, 33*(4), 964 - 977.

Lantao, L., & Sukhatme, G. S. (2018). A solution to time-varying Markov decision processes. *IEEE Robotics and Automation Letters, 3*(3), 1631-1638.

Lobli, S., Meyer-Delius, D., & Pfaff, P. (2013). Probabilistic time-dependent models for mobile robot path planning in changing environments. *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5545-5550). IEEE International Conference on Robotics and Automation (ICRA).

Loibl, S., Meyer-Delius, D., & Pfaff, P. (2013). Probabilistic time-dependent models for mobile robot path planning in changing environments. *In International Conference on Robotics and Automation (ICRA).*

Lorenz, E., Hurka, J., Heinemann, D., & Beyer, H. G. (2009). Irradiance Forecasting for the Power Prediction of Grid-Connected Photovoltaic Systems. *IEEE Journal of selected topics in applied earth observations and remove sensing, 2*(1), 2-10.

Maes, P. (1990). Situated agents can have goals. *Robotics and autonomous systems, 6*(1-2), 49-70.

Maes, P., & Rodney, B. A. (1990). Learning to Coordinate Behaviors. *AAAI, 90*, 796-802.

Makridakis, S., & Hibon, M. (1979). Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society: Series A*, 97-125.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3).

Marthi, B. (2013). Robust Navigation Execution by Planning in Belief. *Robotics: Science and Systems.* Sydney.

Martinson, E., Stoytchev, A., & Arkin, R. C. (2002). Robot behavioral selection using q-learning. *RSJ Conference on Intelligent Robots and Systems.*

McFarland, D., & Bösser, T. (1993). *Intelligent behavior in animals and robots.* Mit Press.

McKenzie, E. D. (1984). General exponential smoothing and the equivalent ARMA process. *Journal of Forecasting, 3*(3), 333-344.

Mehra, R. K., & Peschon, J. (1971). An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica, 7*(5), 637-640.

Mitsou, N. C., & Tzafestas, C. S. (2007). Temporal occupancy grid for mobile robot dynamic environment mapping. *Mediterranean Conference on Control & Automation.*

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing atari with deep reinforcement learning.* arXiv:1312.5602.

National Solar Radiation Database. (n.d.). *NSRDB Data Viewer*. (National Renewable Energy Laboratory) Retrieved August 2019, from https://maps.nrel.gov/nsrdb-viewer

Neubert, P., Sünderhauf, N., & Protzel, P. (2015). Superpixel-based appearance change prediction for long-term navigation across seasons. *Robotics and Autonomous Systems*, 15-27.

Oatley, K. (1974). Circadian rhythms and representations of the environment in motivational systems. In D. J. McFarland, *Motivations control systems analysis* (p. 523). Oxford, England: Academic Press.

O'Brien, M. J., & Arkin, R. C. (2017). Modeling Temporally Dynamic Environments for Persistent Autonomous Agents. *Florida Artificial Intelligence Research Society Conference.* Marco Island, Florida.

O'Brien, M. J., & Arkin, R. C. (2019). Adapting to environmental dynamics with an artificial circadian system. *Adaptive Behavior*(https://doi.org/10.1177/1059712319846854).

Paranjpe, D. A., & Sharma, V. K. (2005). Evolution of temporal order in living organisms. *Journal of Circadian Rhythms, 3*(1), 7.

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control engineering practice, 11*(7), 733-764.

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. *In Computational Intelligence in Ensemble Learning (CIEL).*

Roenneberg, T., Daan, S., & Merrow, M. (2003). The art of entrainment. *Journal of Biological Rhythms*, 183-194.

(n.d.). *Safe and Effective Pesticide Use, A handbook for commercial spray operators.* Adelaide: EPA South Australia.

Sakurama, K., Verriest, E. I., & Egerstedt, M. (2015). Effects of insufficient time-scale separation in cascaded, networked systems. *American Control Conference (ACC)* (pp. 4683-4688). 2015: IEEE.

Santos, J. M., Krajnik, T., Fentanes, J. P., & Duckett, T. (2016). Lifelong information-driven exploration to complete and refine 4-D spatio-temporal maps. *IEEE Robotics and Automation Letters, 1*(2), 684-691.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks, 61*, 85-117.

Shim, D. H., Kim, J. H., & Sastry, S. (2003). Decentralized nonlinear model predictive control of multiple flying robots. *IEEE International Conference on Decision and Control* (pp. 3621-3626). IEEE.

Tang, S., Tang, G., & Cheke, R. A. (2010). Optimum timing for integrated pest management: modelling rates of pesticide application and natural enemy releases. *Journal of Theoretical Biology, 264*(2), 623-638.

Tinbergen, J., & Daan, S. (1980). Young guillemots (Uria lomvia) leaving their Arctic breeding cliffs: A daily rhythm in numbers and risk. *Ardea*, 96-100.

Tractica Research. (2017, July 5). *The Robotics Industry Will Reach $237 Billion in Revenue Worldwide by 2022*. (Tratica) Retrieved January 2018, from https://www.tractica.com/newsroom/press-releases/the-robotics-industry-will-reach-237-billion-in-revenue-worldwide-by-2022/

Velayudhan, L., & Arkin, R. C. (2017). Sloth and Slow Loris Inspired Behavioral Controller for a Robotic Agent. *2017 IEEE Int. Conf. on Robotics and Biomimetics.* Macau, China.

Visinsky, M. L., Cavallaro, J. R., & Walker, I. D. (1994). Robotic fault detection and fault tolerance: A survey. *Reliability Engineering & System Safety, 46*(2), 139-158.

Wang, D., Wang, Z., Li, G., & Wang, W. (2016). Distributed filtering for switched nonlinear positive systems with missing measurements over sensor networks. *IEEE Sensors Journal, 16*(12), 4940-4948.

Wang, W. C., Chau, K. W., Cheng, C. T., & Qiu, L. (2009). A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. *Journal of hydrology, 374*(3), 294-306.

Wang, Y., & Boyd, S. (2009). Fast model predictive control using online optimization. *IEEE Transactions on control systems technology, 18*(2), 267-278.

Wawerla, J., & Vaughan, R. T. (2008). Optimal Robot Recharging Strategies For Time Discounted Labour. *Proc. of the 11th Int. Conf. on the Simulation and Synthesis of Living Systems*, (pp. 670-677).

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., & Theodorou, E. A. (2017). Information theoretic MPC for model-based reinforcement learning. *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1714-1721). IEEE.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting, 14*(1), 35-62.