Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Comparative Analysis of Techniques for Spatio-Temporal World Modeling

Ethan Oswald Massey

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Erwin Prassler
M. Sc. Argentina Ortega Sáinz
M. Sc. Sebastian Blumenthal

January 2019

Abstract

Currently, a variety of methods exist for creating different types of spatio-temporal world models. Despite the numerous methods for this type of modeling, there exists no methodology for comparing the different approaches or their suitability for a given application e.g. logistics robots. In order to establish a means for comparing and selecting the best-fitting spatio-temporal world modeling technique, a methodology and standard set of criteria must be established. To that end, state-of-the-art methods for this type of modeling will be collected, listed, and described. Existing methods used for evaluation will also be collected where possible. Using the collected methods, new criteria and techniques will be devised to enable the comparison of various methods in a qualitative manner. Experiments will be proposed to further narrow and ultimately select a spatio-temporal model for a given purpose. An example network of autonomous logistic robots, ROPOD, will serve as a case study used to demonstrate the use of the new criteria. This will also serve to guide the design of future experiments that aim to select a spatio-temporal world modeling technique for a given task. ROPOD was specifically selected as it operates in a real-world, human shared environment. This type of environment is desirable for experiments as it provides a unique combination of common and novel problems that arise when selecting an appropriate spatio-temporal world model. Using the developed criteria, a qualitative analysis will be applied to the selected methods to remove unfit options. Then, experiments will be run on the remaining methods to provide comparative benchmarks. Finally, the results will be analyzed and recommendations to ROPOD will be made.

Acknowledgements

I would like to firstly thank my advisors who have enabled me to achieve this landmark in my academic career. Without their guidance and encouragement this project would not have been possible.

To my friends and family, I extended a warm and heartfelt thank you. Your emotional support was invaluable. Specifically, I'd like to mention my girlfriend, Megan, for her love and continued understanding while pursuing this work. Lastly, a special thank you to Dylan and Megan for your time spent proofreading and correcting the earlier revisions of this work.

Special thanks to Dr. Tomas Krajnik, not only for your contributions to the field, but to this work specifically. Thank you for providing access to an early version of your work and its respective code. Your help, warmth, and friendliness won't be forgotten.

Finally, I'd like to extend my thanks to all my educators, past and present, who have contributed to my academic and professional career.

Contents

List of Figures

List of Tables

1

Introduction

A robot's world model is its internal representation of the environment that allows it to reason and make decisions. This ability to make decisions, and thus how well a robot performs in an environment, can be directly correlated to the quality of the world model that a robot contains. Historically, these models have been static two or three-dimensional representations, but within the past two decades multiple methods have been developed to introduce an additional dimension to these maps, the dimension of time. The inclusion of time allows for a robot to make decisions about when it may want to accomplish a task, or to avoid a certain area at a given time. In the simple case of an office, a model may suggest to avoid areas of high traffic around the cafeteria during lunch hours or that a shortcut between two buildings is open, but only during the work day. These examples illustrate the type of knowledge and efficiency that can be gleaned by introducing a temporal component to a robot's world model. This new type of world model has come to be known as a "spatio-temporal world model", as it is a model of the world that contains spatial information (the physical environment) and temporal information (how the environment changes through time).

Automatically guided vehicles (AGVs), now common in the field of logistics, stand to benefit a great deal from these improvements in world modeling. Moving from point A to point B is an extremely common task in a wide variety of domains including industrial, commercial, and residential applications. Thus far, AGVs have primarily been used in industrial settings but have been relegated to a discrete

and limited set of predictable tasks. This is especially true when logistics involve a particularly dynamic or human environment.

Recent work into introducing a temporal component to world models has already begun and shows great promise. A variety of methods have been introduced to allow for an AGV to observe and make predictions about its environment through time. However, since this field is relatively new, and with new advancements and approaches being introduced yearly, it is increasingly difficult to evaluate or choose between the different spatio-temporal world modeling options. It is for this reason that a method, or set of criteria, be devised for comparing and contrasting the variety of solutions. The analysis of these methods will not only allow for others to choose the most fitting approach for a given environment, but also expose deficiencies in the current approaches and guide future research efforts. Improvements in the field will ultimately result in more flexible AGVs that can operate in a wider variety of environments and for longer periods of autonomy.

1.1 Challenges and Difficulties

Historically, world modeling techniques can be thought of as simply a mapping and path planning problem in either two-dimensional or three-dimensional space. These problems have been studied for decades and thus there already exist a handful of well-known solutions, each with its own advantages and disadvantages. However, with the recent inclusion of the fourth dimension, time, a number of different methods have been introduced.

The early and simplistic approaches to introducing temporal components into world models started as early as 2002 [3] but within the past two decades there has been an increase both in the number of different approaches and the complexity of the methods. This combined with a lack of historical perspective and analysis makes the selection of a best-fitting spatio-temporal world model daunting for new projects. A few papers have introduced some simple means for comparison, but these have been limited to a rudimentary discussion about the space and time complexity of an approach, or an internal evaluation of different tuning parameters of a proposed method.

1.2 Motivation

With numerous different methods and no historical knowledge or criteria of comparison this paper aims to provide a template for comparing existing models that should be extensible to account for the inevitable release of future methods. To that aim the following goals shall be met:

- Summary of the major existing spatio-temporal world modeling techniques.
- Collection of performance measurement or other comparison techniques as defined by the papers themselves.
- Introduction of metainformation in order to better compare the existing world modeling techniques
- A quick and easy-to-use table for high level overview and comparison of techniques
- Example application of the aforementioned information to select a fitting technique for a real-world application
- Design and execution experiments to further test the selected techniques
- Subsequent evaluation and discussion on the appropriateness of technique selected, especially with respect to the real-world use and introduced metainformation

It is with this collection of existing comparison techniques, new metainformation, and a tangible example that other projects may be able to more easily evaluate and select the best fitting spatio-temporal world modeling technique for the project. Additionally, when new spatio-temporal world model techniques are introduced, it should be with relative ease that their information be integrated into this method for comparative analysis for future use.

1.3 Problem Formulation

In order to best choose between preexisting solutions for spatio-temporal world modeling and guide future development, it is vital that comparative criteria be established. This comparative analysis will set out to clarify and quantify these approaches. Although the comparative analysis will be general enough to be applicable to any project seeking to incorporate spatio-temporal wold modeling, it will be viewed through the lens of a specific real-world application with a focus on long-term planning. Additional details about the specific application will be discussed later, but it is important to note that viewing the various modeling methods through this lens of a real-world application is quite powerful.

Improvements in world modeling, specifically within the domain of spatio-temporal world modeling, have already yielded significant increases in the performance of robotic logistic systems. These improvements led to decreases in travel time as well as increases in reliability, which hinge on knowing what areas to avoid at which times. In turn, these improvements create a much more powerful and scaleable logistics network with less downtime. This results in more goods being delivered, savings of both time and money, and, especially in the case of a hospital, freeing up previously busy employees for more demanding work.

Despite these benefits and the large number of existing approaches for spatio-temporal world modeling, there is currently no method for accurately comparing and contrasting multiple approaches for a given task. It is with this in mind that this report will collect, describe, compare, and contrast these approaches. It will use criteria already available, when possible, as some of the work already includes basic performance statistics. Furthermore, in work where these criteria are not mentioned explicitly, or are not otherwise available, an attempt to derive the information will be preformed either via calculation or collected via simulation. Lastly, new criteria will be devised or otherwise assigned to account for information desired and not provided or other metainformation that would aid in comparing these methods.

Finally, an example study will be included that will attempt to determine the best-suited approach for a real-world scenario, known as ROPOD. This scenario in

question involves optimizing the logistics of moving supplies or bed carts internally within a busy hospital. It consists of a central server in charge of planning and routing multiple robots. Additionally, it will be assumed planning will be done with OpenStreetMap and thus will use a graph-based approach. More details and specifics about this project will be discussed in a later section.

2

State of the Art

Given the variety of different approaches to for implementing a spatio-temporal world model, the methods have been divided into groups. A significant number of spatio-temporal world models are implemented on top of preexisting world modeling techniques and are therefore tied to a specific spatial representation. There are exceptions to this, however, with some models being built from the ground up, intertwining spatio and temporal components.

2.1 Map Dependent Models

2.1.1 Occupancy Grids

Occupancy grids were first introduced in 1985 by Moravec and Elfes[1] . In simple two-dimensional terms, they can be thought of as a grid placed over an environment. Each cell then represents the probability or belief that a given cell is either occupied or free. Free, in this case, meaning that a robot would be able to traverse through that cell. This concept can of course be extended into the third dimension for a more complex world model.

Temporal Occupancy Grids

One of the earliest and most straightforward attempts to introduce a temporal component was to extend the existing world model in question, in this case occupancy grids. This can be seen in [3]. In this paper, Arbuckle et al introduce the concept of "temporal occupancy grids" (TOGs). The authors noted that the key to these TOGs was that they "can differentiate between different patterns of occupancy, even when the absolute probability of occupancy is the same." That is to say, one could imagine a parking lot where it would be possible with TOGs to distinguish between cells that are parking spaces, cells that are pathways, and cells that are not for driving at all, such as a median. These TOGs also made it possible to detect where a door or elevator may be.

Temporal Occupancy Grids were developed by generating multiple occupancy grids with contemporary methods but each occupancy grid would represent, and be generated using samples from, multiple different timescales. With multiple occupancy grids spanning multiple timescales, the probability of a cell being occupied could be computed by a simple summation.

Hidden Markov Models

Hidden Markov Models (HMMs), are a type of Markov Chain that can be considered [9]"a doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations.". Simply put, an HMM can be thought of as having N number of states S, that are hidden, or otherwise not directly observable. Each state can have M number of observations made about the properties of that state which may reflect indirectly, and to varying degrees of certainty, the actual state. Furthermore, each one of these states has a given probability distribution of transitioning from one state to another. It is from this information that a Markov Model or Markov Chain can be constructed.

In the specific case of occupancy grids, each cell can be thought of as having two

states, either free or occupied. It is not feasible to directly observe every given cell at all times, or more specifically, at the time of path planning. Thus their states are considered hidden. However, through past observation and data collection, a cell's data can be considered to be known throughout time. Thus, this temporal data can be thought of as observational data and can be used to make predictions about state transitions.

Early combinations of HMMs with occupancy grids differed from previous dynamic world modeling approaches as this approach [4] "does not depend on dynamic object detection and high-level object models; it considers only the occupancy of the space at a lower level of abstraction". By relying on and collecting lower, more easily observable data, larger amounts of data could be collected and processed over greater periods of time. Since each cell is dependent only on previous observations of that cell throughout time, the increase in data availability and the discrete nature of the predictions led to improved state predictions.

Building on this work, Meyer-Delius [4] introduced the concept of online learning to the HMM approach. Traditionally, offline learning has been used when a robot's navigational system would hold a copy of a world model produced some time before operation. However, it is possible that objects in the robot's environment may have changed between the time the map was generated to the time at which the robot operates. With the introduction of online learning, the robot would be able to observe these changes and factor them into its existing navigational system. This was the first development that attempted to surpass the static nature of the transition states of the HMM model.

In [20] a further improvement to occupancy grids with HMMs came with the introduction of being able to model trajectories of the objects in an environment. Although early work on this concept was done without the use of HMMs [19], newer techniques to the same end have been developed using HMMs. This new approach was an important improvement because the dynamic motion of objects in an environment, such as humans walking in a hallway, could now be better modeled. This process was dubbed "Input-Output HMM" (IOHMM) due nature of how cells

in the grid would communicate with one another. Each cell would look through it's own historical data, but would also communicate with its neighbors. In effect, this allowed a cell in hallway to be able to predict occupancy based off of a nearby cell that is currently occupied as well as its historical occupation. This is particularly well suited for the prediction of trajectories and resulting cell occupation.

2.1.2 Spatio-Temporal Hilbert Maps

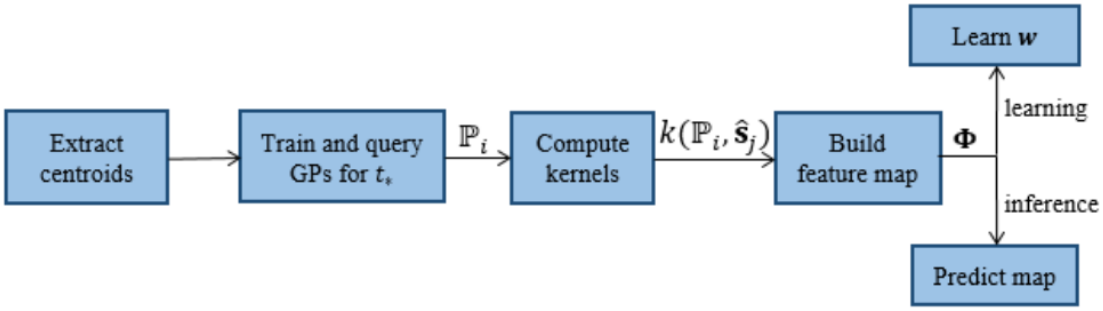

Figure 2.1: Spatio-temporal Hilbert map training process (GP - Gaussian Process) [12]

In contrast to the discrete nature of occupancy grids, Hilbert maps provide a continuous representation of an environment which allows for arbitrary world model resolution. They rely on "fast kernel approximations that project the data in a Hilbert space where a logistic regression classifier is learnt". A stochastic gradient optimization can then applied. This approach is similar to that of a Gaussian processes occupancy map but with a much lower computational cost. An example of the training process workflow is visible in figure 2.1. A new development as of 2016 in [6, 12], Hilbert maps with the addition of a temporal component are a new field of research, to which the authors of the original paper are still contributing.

Further insights is provided in [12] [13] where they mention that in static Hilbert maps, the kernel can be thought of as the location of an obstacle or object. When

introducing the temporal dimension, the centroid of a moving object is extracted from raw data over time. This data can the be used and trained on to create a model that can predict both the direction and speed of an object at a given location at a given time. It is particularly well-suited to short-term predictions such as car traffic on a road or at an intersection.

2.2 Map Independent Models

2.2.1 Multi-Map Approach

Extending or perhaps drawing inspiration from some of their earlier work such as [3], a map-independent approach was introduced by Biber, et. al. in the mid-2000s. In their paper "Dynamic Maps for Long-Term Operation of Mobile Service Robots" the authors describe a method for predicting and anticipating changes in a dynamic environment to improve localization and navigation. This was done by maintaining multiple maps of a given area, with each map being updated at a different time scale. Specifically, each cell in an occupancy grid or each edge in a graph would have an author-determined number of maps associated with it. In the paper, the example uses three maps with the idea being a short-term, medium, and long-term memory/map of an environment. Each of these maps are given a set number of observations, e.g. 20. When the desired number of observations have been made, all the maps are updated. When an update happens, a certain number of observations are selected at random to remain and the map prepares for the next set of observations. The number of these observations to remain, sometimes referred to as the update ratio, determines what timescale a map will model. Maps that keep few observations are said to act like a "short-term memory" with "long-term memory" maps maintaining most of their observations during updates. This allows a single map to be chosen at a given time or for multiple maps to be averaged together, depending on the desired behavior.

This method's primary benefits are the simplicity of design and the ability to customize or tune the method during operation. This is true regardless of pre-existing map implementations. Unfortunately, a large amount of customization and tuning

opportunities comes with a cost. Tuning parameters can be a tedious and time-consuming process. Additionally, parameters that work well for one problem may not work well for another. Finally, the authors note that the method for selecting the correct map is not an easy problem and often depends on the environment encountered and the data collected.

2.2.2 FreMEn

Frequency Map Enhancement (FreMEn)[8] , is a technique for spatio-temporal world modeling that can be used independent of mapping or world modeling technique. Introduced in 2014, its original aim was to improve mapping for long-term scenarios. They observed that many previous approaches focused on mapping multiple static environments over time, which suited slow-changing environments, but may not work well for dynamic settings. In response to these issues, the authors set out to design a method who's original goals were to improve mapping for long-term scenarios and to generate world models for highly dynamic environments. Although FreMEn was initially used with octomaps, three-dimensional occupancy grids, it was later decoupled from this mapping technique, allowing for a more broad and flexible technique.

In its original and most basic form, FreMEn assumes that an environment can be broken down into multiple independent components. It is then further assumed that these independent components will take one of two binary states. Examples of this include a door being open or shut or a cell in an occupancy grid being either free or occupied. Each of these states can not always be directly observed, and the tools e.g. sensors available to the robot my introduce noise and cannot be taken as one hundred percent accurate. Thus, each component has a certain probability assigned to it. This probability defines the likelihood of that component being in a given state, e.g. a door open or closed. Since these states can be observed multiple times over a given period, their probabilities can be defined as functions dependent on time.

Once in the time domain, FreMEn can use a well-known mathematical tool

commonly used for signal processing, the Fourier Transform. Since FreMEn focuses on long-term observations of dynamic, often human, environments, it is assumed that harmonic patterns will develop over time. Applying a Fourier Transform to the data collected it is possible to describe the observed data as a series of periodic functions, each with different periods and weight coefficients, summed together. The top l most influential/fundamental frequencies, those with the largest coefficients are selected for storage to describe the behavior. l is also know as the order of the model. It allows the model to generalize the predictions which prevents over fitting and avoids storing excessive information and increased complexity. In order to make predictions, the l functions are summed together at a given time t. Furthermore, because the Fourier Transform is reversible using the inverse Fourier Transform, one can easily convert between the stored observations in the spatial domain back to the time domain. This allows for predictions at any given time t. Not only is this useful for future predictions, but this method can also be used to analyze the accuracy of the model by comparing previously observed data from the past to the model's actual predictions. Using this historical accuracy, one can then tune the order of the spatial model to obtain more accurate historical predictions with hopes of also having more accurate future predictions. More information on this process can be found in the original paper [8] and the follow up FreMEn paper [18].

Improvements and Additions

As ground breaking and flexible as FreMEn is, due to the many assumptions it makes, it is not without its flaws. One major assumption made is that areas of observation will be observed not only frequently, but periodically in the most strict sense. That is to say, it is not only important that a location be visited and observed, but that the observations follow a pattern that is temporally regular and consistent. This requirement is imposed by the use of the Fast Fourier Transform (FFT) technique as described in the original papers. To address this, later authors devised and implemented other methods of storing data and making predictions. These methods in [7] often involve phase shifting, modifying the amplitude of the observation, or using a modified equation derived from the Fourier Transform instead of the standard FFT.

Another major limitation of FreMEn is its assumption that all observable behavior can be modeled with binary states. One attempt at solving this issue involves replacing the Bernoulli distribution of FreMEn with a Poisson base approach. The aim of this being to better represent human patterns present in an environment such as an office building or hospital. Specifically, as mentioned in [5] they "extend the technique (FreMEn) by employing both Poisson processes as the counting model to replace the binary states of FreMEn and a new way of selecting the most prominent frequency components of the Fourier spectrum." This approach, however, does not come without its own set of assumptions and problems. By using a Poisson process instead of a Bernoulli distribution, predictive models are no longer limited to binary states. Because a probability mass distribution is used, the primary factors are γ and N, γ being the average number of occurrences of an object or behavior in a fixed time and N being the actual number of occurrences in a given time frame. These factors limit this approach to only being able to estimate objects or behaviors that are countable. For instance, this would not work well for something like predicted travel time.

In the same paper, [5], another interesting modification was made. In order to mesh the Poisson Process with the observed month long data and also increase confidence, the data was broken into week-long segments and then stacked on top of itself. The thought behind this is that it not only increase the number of data points to train on, but also accounts for behaviors in human environments that can often be cyclic on weekly boundaries. Although this works well assuming all behaviors happen with weekly periodicity, that is not always the case, and thus behaviors that don't occur every seven days at the same time may be incorrectly predicted. An example of a week's worth of this data is visible in figure 2.2.

The idea of manipulating and folding observed data over itself through time continued to evolve. Recently, in [17] the idea was taken a step further. This paper, continues to use FreMEn at its core. Meaning it uses the same Fourier-like approach to observing periodicity in a dataset. It also continues to build off of the ideas mentioned in [5]. Now, no longer is data restricted to being folded at a single point,

instead, it is folded back on itself at one of many possible frequencies. When and where to fold the data is determined using one of two recommended methods: either expectation maximization or K-means clustering. The processes of finding when and where to fold is iterative and the number of folds, which can be thought of as the order, can be either hard limited or determined during training when the error introduced by increasing the order is greater than that of the previous step. This approach not only allows for non-binary predictions of any data observation type, but also for sparse or infrequent data to be the basis of prediction, due to the folding of the data.

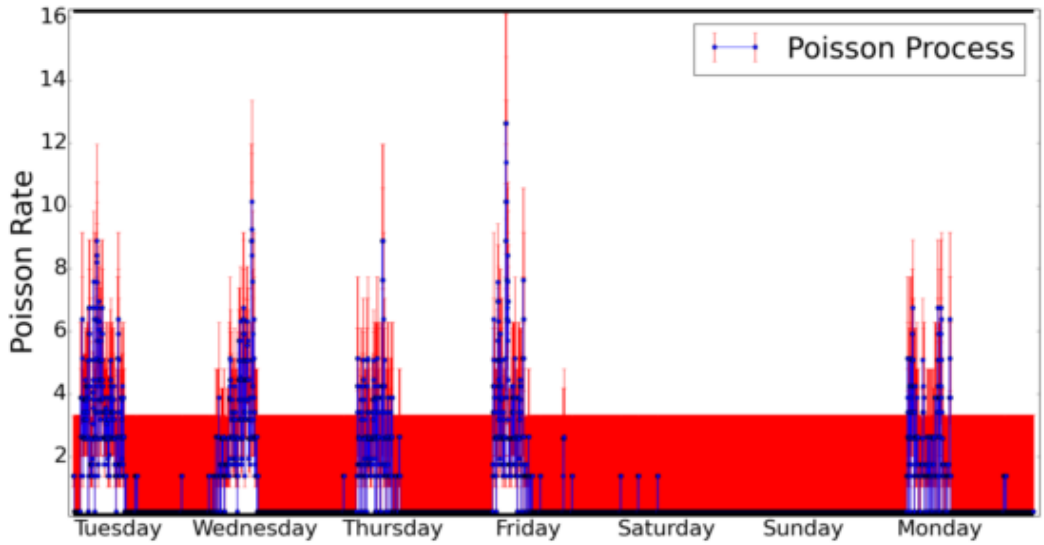

Figure 2.2: Lambda time series of a corridor using Poisson Process [5]

2.3 Existing Methods for Evaluation or Comparison

It is clear that there are a multitude of different methods for creating spatio-temporal world models. Given the wide range of methods available, there has been an initial push to create a means of evaluating and comparing methods with other approaches. The vast majority of these evaluations look internally, however. They compare a given method that uses a fixed set of parameters with the same method

but with parameters slightly varied from the original. Yet others, although fewer in number, do set more general criteria that can be directly used or extended to evaluate their performance with respect to any other spatio-temporal world modeling technique. Because methods for comparing a model against itself with different parameters can often be very specific, this section will focus on those existing methods that have been, or can be, extended to allow any modeling method to be compared to another.

2.3.1 Current Methods and Limitations

Prediction Accuracy

Arguably the most obvious and most used metric for comparing spatio-temporal world modeling techniques is to compare the accuracy of a prediction to the ground truth over time. In the case of binary data, for example in [4] the accuracy of an occupancy grid map is evaluated as a whole over a discreet number of time steps. For a given time step t, each cell in the grid is compared to a ground truth value. These are then summed up to produce an average accuracy at a given time t and plotted over time. These were used to compare different maps produced by the same algorithm but with different parameters. "Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments"[12] takes this a step further and advances this method by comparing variations both forward and backward in time. This allows for further improved training and adjusting tested technique's parameters. Finally, "Learning Temporal Context for Activity Recognition" [2] extended this method for comparison to be used to compare multiple different spatio-temporal world modeling techniques over multiple days worth of data.

Resource Usage

One relevant method that is often used in other fields of study is a comparison of run-time usage of computational resources. This is even more pertinent when considering the effect spatio-temporal world models may have on a system when

deployed to a real-world environment. Particularly, this is especially important when the ability to scale is required. Unfortunately, there is currently no standard technique for collecting or comparing this data. Perhaps one of the only examples of a discussion of run-time resource usage appears in [12]. Sadly, this is only gives a rough estimate, on the order of a half second, that an average single observation and prediction would take. Moving forward it would be desirable to have not only detailed observations of run-time memory and processor usage, but also a comparison of resource usage with respect to the multiple techniques being compared.

3

Criteria for Comparison

Given the wide variety and complexity of the methods available for spatio-temporal world modeling it's vital that criteria be established in order to properly and efficiently compare them. As seen in the previous section, some initial work has been done in this regard. This section, however, aims to introduce a complete list of criteria that can be used for comparison. The methods will be introduced, outlined, and their relevance to the field explained. The idea is to provide an overview of some common techniques for modeling and to establish a way to easily compare future spatio-temporal world modeling techniques.

3.1 Criteria

3.1.1 Computational Complexity

According to Michael Stiff, a professor at the University of Wisconsin-Madison, computational complexity can be defined as "the study of how much of a given resource a program uses. The resource in question is usually either space (how much memory) or time (how many basic operations)." [10]. In the field of computer science this computational complexity is often represented using asymptotic notation, also commonly known as Big O notation. The general idea is to estimate how much of a given resource will be used as the size of the input data is increased, specifically as it

approaches infinity. This is useful for estimating how much memory or how much time a given algorithm may use for a given set of data. Additionally, it serves as an excellent benchmark for comparing various algorithms.

In the particular case of spatio-temporal world modeling, one is most concerned with how long planning will take and how much memory the process will consume while planning. Furthermore, although Big O notation is still applicable and accurate within reason, some details may be lost. Specifically, Big O will often drop coefficients in growth rates as they are not relevant when looking at asymptotic behavior. An example of this with regard to spatio-temporal world modeling would be if one model stored a single copy of every observation made while another stored multiple, say five. Although the second model would grow five times as fast as the first, according to Big O notation, they would both be linear, thus growing at the same rate of "N". This is obviously a drawback when the vast majority of modeling techniques would all be of a similar complexity class for both time and space. For the purpose of comparative analysis outlined in this paper, Big O notation will still be used. However, in order to highlight subtleties between mapping methods, coefficients in growth rates will not be dropped.

3.1.2 Method of Data Storage

Similar to spatial and time complexity, the method of storage also evaluates how a method handles the resources available to it. However, where spatial and time complexity are focused on the use of resources during run-time, the evaluation of the method of storage is focused on how well and in what manor the information used for predictions can be stored when not in use. This can be important when looking at the scaleability of a particular model. One can imagine if hundreds of thousands of observations are being made every day, depending on the data contained within those observations, an inefficient storage approach could rapidly cause giga- or terabytes of storage space to be required. Therefore, it is important to note once learned or trained, if and how information and predictions can be stored.

3.1.3 Learning Method

The learning method is a way a to describe how a spatio-temporal world model is able to make predictions. Often this is based off of preexisting concepts like neural nets or other mathematical models. Information on these methods is commonly known and thus additional information may be desired to further compare methods. One such vital piece of information is distinguishing when a model changes or updates in relation to when predictions are made. A common way to make this differentiation is by categorizing an algorithm or method as one two categories: Online or Offline. Offline algorithms methods have been the historically common approach. Data is taken in an analyzed in batches before any predictions are made. Online learning on the other hand sees a much quicker turn-around time between the observation of new data and the resulting prediction [14]. For the sake of this paper, methods will be classified as online if they update after every new observation or after a given number of observations. Methods will therefore be classified as offline if they are designed to process large amounts of data before making predictions, specifically if processing time makes live and immediate predictions unreasonable.

3.1.4 Map Type Dependency

As mentioned in the State of the Art section, certain spatio-temporal world modeling methods have been developed with specific world modeling techniques in mind. It is quite probable that these different modeling methods would have an impact on the performance of prediction. Additionally, from a developer stand point, it may be desirable to extend a preexisting world model with the ability to make spatio-temporal predictions. It is therefore conceivable that an environment may already be modeled using a specific technique and thus limiting the pool of potentially available methods. For these reasons being able to differentiate, and thus choose between, the various methods with relation to the world modeling method required is highly desired.

3.1.5 Limitations on Predictions

When examining a large number of options it is often helpful to eliminate choices that absolutely will not work. This section will attempt to outline the types of limitations on predictions or model representation that would disqualify certain spatio-temporal world models when attempting to find a best fit solution for a give problem.

Classification Restrictions

Certain techniques used for predictions can cause limitations on the number or type of classifications that can be done on a given set of data. The most common and perhaps most limiting is the binary restriction imposed by the methods that use FreMEn [11, 18]. This restriction only allows an object's state in the spatio-temporal domain to be represented by one of two states. This state is then said to have a probability associated with it. Thus, it is often not possible to represent more complex behavior like time of travel, or the number of people present in an area, which often requires the use of more than two states.

Historical Predictability

The ability to reflect on and learn from previous mistakes is as useful for humans as it is in robotics. Being able to recreate and analyze historical events allows for the unexpected to be debugged and for a model to be improved through iterative training. These improvements can ultimately result in more accurate predictions. Therefore, the ability to predict events that have already happened can be a desired trait when looking at spatio-temporal world modeling techniques.

Long-Term Predictability

Conversely, compared to historical predictions, sometimes it may be desirable to make long-term future predictions about the world. That is to say, predictions not just about what is immediately going to happen, but about what may happen in the short, near, or long-term. Many scenarios can be imagined in which this would be advantageous. Long-term future predictions allow for increased autonomy and decreased requirements for offline learning. Furthermore, long-term predictions about the future can also be used to retroactively evaluate the performance of multiple models, perhaps using the same technique but with different parameters.

3.1.6 Metainformation

Some information about a spatio-temporal world modeling technique may not be directly observable or even contained within the approach itself. Information about how and when a method is best used or other metainformation not directly tied to an approach, can be vital to a developer when attempting to decide between various methods. This section introduces some of the metainformation about spatio-temporal world modeling techniques that could prove useful, especially with regard to a developer or engineer.

Availability of Work

Commonly, during or after the development of a new spatio-temporal world modeling technique the author or authors will release the new method in the form of a library or source code. The availability of libraries or source code can save time developing and debugging. Additionally, the preexisting work may have an effect on what language will be used on a given project. The availability of work can be categorized into three fields. Open source means that the work is freely available as a package or source code. Partially available means that some of the subcomponents may be freely available, but not connected to create the entire package, or the component is available, but closed source. In this case, it is common that the connecting or

support structures are not yet implemented and it would take some time to connect everything together. Finally, some work may not currently exist in any available form. One would have to fully acquaint themselves with the original academic paper in order to implement this method.

Suitable Fields of Application

Although the field of spatio-temporal world modeling is generally concerned with how, when, and where things are happening, not all methods attempt to predict or model the same behavior. It is often common for a given modeling technique to be developed for a specific problem. Sometimes it is possible to modify these approaches to fit other fields. Other times, generic approaches are developed directly. This section aims to categorize how specific a method is for a given problem classification and how well suited is for certain sets of problems.

3.2 Experimental Criteria

Comparing and contrasting various techniques on paper is an excellent way to narrow down one's options when choosing a modeling technique. However, at a certain point it is desirable to evaluate the remaining methods with respect to performance data. This information cannot always be obtained via research alone due to lack of information provided in the original or subsequent research. Additionally, results from one experiment in a given domain may not directly translate to that of another. Therefore, in order to best compare and thus most effectively choose a technique for spatio-temporal world modeling, some experiments must be done. The following criteria are a recommended starting point for comparison, however, the specifics of a project may demand modified or additional analysis.

3.2.1 Prediction Accuracy

Prediction accuracy is arguably the most important benchmark for comparing methods of spatio-temporal world modeling. It can take many forms, but the goal is

to convey on average how well a model can predict a future event. This is most commonly achieved by comparing a predictive model to the ground truth data. For binary data, this is often simply done by counting the number of incorrect predictions. Non-binary data becomes a bit more difficult and, depending on the data being worked on, there are multiple approaches. The easiest to convey is calculating and tracking the average distance from a correct result. Regardless of the data being tracked, the end result is best viewed in the form of a graph over time. This allows for the most concise comparison of the performance of various spatio-temporal world modeling techniques.

3.2.2 Computational Resource Usage

Computational resource usage can be thought of as an extension of computational complexity into the real-world. This statistic aims to quantify and measure the time, memory, or any additional resources a computer provides during run-time. Most commonly these resources are the amount of time spent using a processor and the amount of memory used. These are commonly measured using seconds and bytes in a resolution scaled to the size of the problem, milliseconds and kilobytes for example. These are excellent ways of predicting the ability of a given method to scale. In a real operating environment it does not matter if 100% accuracy is obtained for all datasets if too much memory is required for a computer to process the results or if the time taken to produce the prediction invalidates the very prediction it is attempting to make.

4

Qualitative Comparative Analysis

With a large number of methods for spatio-temporal world modeling outlined in the State of the Art section, this section will apply the comparison tools to an example case study. For the purposes of this paper the ROPOD project has been selected as the candidate for the case study.

4.1 What is ROPOD?

ROPOD is a research project funded by the "European Union's Horizon 2020 research and innovation programme" that aims to develop "ultra-flat, ultra-flexible, cost-effective robotic pods for handling legacy in logistics"[1]. In additional to its business oriented goals, ROPOD, being a research project, also has a focus on the development and implementation of cutting edge technology. The main goal of the project is to enable low cost robotic solutions for coordinating logistics in a preexisting human environment. Specifically, the current target testbed for ROPOD is a mid-sized hospital in Frankfurt, Germany. ROPOD is comprised of a multi-robot system combined with a central server. Although the individual robots are in charge of navigation and orientation on a local level, the central server is responsible for planning when and how specific actions are carried out. In particular, the central server uses a modified and extended version of OpenStreetMap[2], also known as OSM,

[1] http://www.ropod.org/
[2] https://www.openstreetmap.org

to maintain the main global world model. This means that all information storage and path planning needs to work in conjunction with OSM. Of specific interest to this project is the global path planning, which focuses on modeling both individual objects in an environment and maintaining a graph with nodes and edges.

4.2 Why ROPOD?

ROPOD offers both benefits and interesting technical challenges that make it excellent for a case study. First and foremost, one of the main benefits is that the robots involved will be operating in a real-world environment with all the intricacies and complexities that accompany it. Of particular note for this case study is that the operating environment is cohabited by humans. It is postulated that many of the behaviors and dynamic obstacles encountered will be of periodic and thus predictable nature. Furthermore, because ROPOD is a multi-robot system, as the number of robots increases, so will the frequency and range of data collected. It is believed that this will further increase accuracy of predictions, or at the very least increase the size and scope of the dataset used for training. Because the targeted area of operation is a known environment with tracked dynamic behaviors this allows for paths to be planned a significant time in advance. Furthermore, the use of a central server allows for more complex calculations and predictions to be done. In total, these benefits allow both a methodical and comprehensive model to be developed.

However, ROPOD is not without its technical complexities. A result of the operating environment being both dynamic and human, many different types of objects and behaviors must be modeled. Binary objects such as doors being open or closed, are obvious examples, but more complex, non-binary objects must also be efficiently and accurately modeled. Complex variables in this case range from the wait time after an elevator has been called, to the average travel time between two nodes, and the density of people in a specific area. All of these behaviors and more must be modeled for any given time and location in the hospital. This provides considerable test space with numerous areas for technical complications. Additionally, although ROPOD is being funded and targeted for research, it still requires efficiency with regard to project development and forces projects to focus on practical results

within a reasonable time frame. This means that even if an algorithm or method is theoretical the best on paper, it may not be the best solution for ROPOD if it consumes to many computational resources or has other requirements that cannot be met. To that effect, any new technology must integrate with existing technology that has already been integrated with the ROPOD system. This means that any spatio-temporal world modeling method chosen must mesh well with OSM and other relevant design elements.

4.3 Methods Under Consideration

Drawing from the spatio-temporal world modeling methods mentioned in the State of the Art section, a handful have been selected for discussion here. The methods selected for additional analysis here were chosen specifically to allow for a variety of modeling techniques to be compared while also allowing for comparison of some similar styles in order to highlight the benefits of a comparative evaluation. They provide both a varied range of methods as well as multiple methods of similar styles that have been included for further in-depth comparisons.

Therefore, the following methods will be evaluated for use with ROPOD:

- Occupancy Grids with Input-Output Hidden Markov Model (IOHMM) - Modeling Spatial-Temporal Dynamics of Human Movements for Predicting Future Trajectories [20]
- Conditional Transition Maps (CTM) - Conditional Transition Maps: Learning Motion Patterns in Dynamic Environments [19]
- Dynamic Multi-Map (DMM) - Dynamic Maps for Long-Term Operation of Mobile Service Robots [15]
- Spatio-Temporal Hilbert Maps (STHM) - Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments [12]
- Frequency Map Enhancement (FreMEn) - FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments [18]

- Hypertime - Warped Hypertime Representations for Long-term Autonomy of Mobile Robots [17]

The following sections contain a brief overview of the selected methods. Although they will not cover all the specifics of a given model, they will mention the relevant information for comparison. Sources have been provided for each paper if the reader wishes to build a deeper understanding of one or more of the approaches.

4.3.1 Occupancy Grids with Input-Output Hidden Markov Model (IOHMM) [20]

As the name implies, this method is primarily inspired by, and meant to be used with, occupancy grids. This effects many of the design decisions and resulting performance metrics. Aside from the regular computational complexity associated with neural networks, the Markov model was designed to be used with 4+1 edges. One for each of the neighbors, up, down, left, and right, as well as one for the center, or no change. Although this is how it was first described in the paper, it is possible to imagine this being extended to any arbitrary number of connections for use with other grids or graphs with more or less than 4 neighbors per cell or node. Despite this possibility, as it is described in the paper, the method is specifically designed with HMM occupancy grids in mind and thus is considered map dependent. Since just the HMM weights can be saved, the data storage size is reasonably low. However, because of how an HMM is used, and the fact that only one model is stored, there are no future or historical predictions meaning that only current predictions are possible. Due to the use of neural nets the learning method is considered offline because significant training must be done for prediction. Finally, although no source code or package is directly available, this method relies heavily on preexisting technology like HMM, occupancy grids, and neural nets. Using the paper as a guide, it would be reasonable to tie these technologies together without too many difficulties, not withstanding the standard roadblocks when working on a new project.

4.3.2 Conditional Transition Maps (CTM) [19]

CTMs, as proposed in "Conditional Transition Maps: Learning Motion Patterns in Dynamic Environments" [19], are similar to IOHMM on a conceptual level. They both involve using the connections between occupancy grid cells to model and predict future behavior. For similar reasons they are unable to do historical or long-term future predictions and are restricted to only predicting tendency of motion over time. The implementation details between the two are, however, different. CTM has additional connections at each cell. There connections for the four cardinal directions from a cell along with intermediates which increases the number of connections from 4 to 8. Furthermore, the direction of travel, in or out, is also considered. Combined, this means that there are 64 parameters per cell, one parameter for each combination of input and output direction. Neural nets are also not used for learning in this and instead a custom designed method is proposed.

4.3.3 Dynamic Multi-Map (DMM) [15]

The dynamic multi-map approach heavily depends on the number of maps used and their respective timescales. Although DMM was originally intended for use with occupancy grids. It can relatively easily be implemented to represent any number of objects or behavior. Additionally, selection of the maps is not defined. The maps themselves being a series of averages means this method acts as an online learning method. Because of this, historical and future predictions are not inherently possible. If maps were stored then historical predictions could be possible but this would also drastically decrease the storage efficiency dependent on the rate of sampling. Finally, although no current implementation of DMM is publicly available, given the relative ease of maintaining multiple maps and keeping a given amount of data during updates, DMM is fairly simple to implement from scratch. The major outstanding drawback is the lack of a well defined method for choosing between, or combining, maps. This oversight is unfortunate as the method for selection or merging the map/maps has a large impact performance metrics of DMM.

4.3.4 Spatio-Temporal Hilbert Maps (STHM) [12]

STHMs extend existing Hilbert maps with the goal of providing real-time dynamic object tracking and prediction. Objects are detected as moving when two or more scans show motion between their estimated centers. These objects are then given motion vectors at every given time t. This means that not only does model complexity scale with the number of dynamic objects, it also scales with desired length of historical observations being made. Historical predictions are possible due to the fact that observations are being kept over time and predictions on objects are also being made over the same time. Furthermore, future predictions can be made with included uncertainty as time increases. This method uses many preexisting technologies, such as Hilbert maps, stochastic gradient descent, and Gaussian process regression. Considering the large and diverse number of techniques and the complexity with which they are connected to make a working system, this approach is considered to have an extremely high implementation complexity. Finally, no open source implementations are available.

4.3.5 Frequency Map Enhancement (FreMEn) [18]

FreMEn relies heavily on the use of a modified version of the Fourier transform and this decision informs a lot of it's properties. A model order must be defined for FreMEn. This order, which can be thought of as the number of cosines with coefficients that will be summed to make a prediction, dictates both time and, to some degree, space complexity for training. Once trained, however, predictions can be obtained with a single, relatively computational inexpensive, equation. Therefore, FreMEn can be viewed as an offline learning method where training is done over a relatively long period of time and a model can be used for a given set of time before training again. Additionally, because predictions are done by computing a result at a given time t, predictions can be made at any arbitrary time, past, present, or future. FreMEn is well suited for doing predictions on any type of map or object with the caveat that the data must be binary, that is data can only be classified into one of two categories. Finally, since the implementation heavily relies upon a commonly

understood concept with only minor support, it is considered a 2 for complexity. This is a non-issue, however, as FreMEn is available both on GitHub[1] as well as in packages available for Ubuntu that add ROS support[2].

4.3.6 Hypertime [17]

Created as an extension to FreMEn, Hypertime does not have a binary restriction on the classification of data. Additionally, data is now dynamically analyzed and wrapped back in on itself to improve observation coverage and improve predictions. Both of these behaviors are heavily dependent on the clustering method, of which Hypertime currently provides two; K-means clustering and Expectation Maximization. Both of these methods are discussed at length in the original paper, but their main goal is determining where and when the data observed should be looped back in on itself. Depending on the order provided and the number of clusters estimated, a linear scaling performance hit is expected. Once again, because Hypertime uses FreMEn, historical and future predictions are possible. Finally, this work is openly available on GitHub[3].

4.3.7 Summary of Approaches

Figure 4.1 provides a summary of the information gathered in the previous sections.

[1]https://github.com/strands-project/fremen
[2]http://strands.acin.tuwien.ac.at/software.html
[3]https://github.com/gestom/Hypertime-RAL-18-0278/tree/master/door_state/src/models

Name	Additional Model Computational Complexity	Learning Method	Offline vs Online	Efficiency of Data Storage
IOHMM	(4+1)*Cells	Neural Net	Offline	HMM weights
CTM	(8*8)*Cells	Cross-correlation	Offline	Cell Connections
DMM	Scales linear with number of maps	Stochastic Sampling	Online	Varies with number of maps and samples per map
STHM	Scales linear with number of dynamic objects being tracked	Stochastic Gradient Descent	Online	2D vector per object at every time t
FreMEn	Scales linear with Fourier Order	Modified Fourier Transform	Online	Order dependent summation of cosines
Hypertime	Scales linear with Fourier Order & further dependent on clustering method	FreMen + time folding	Online	Order dependent summation of cosines

Name	Map Dependency	Historical Predictions	Long-Term Predictions	Work Availability	Suitable Fields of Application
IOHMM	Occupancy Grid	No	No	Partially	Motion prediction
CTM	Grid Based	No	No	Not Available	Motion prediction
DMM	N/A	No	No	Not Available	Very versatile. Works with both binary and continuous observations
STHM	Hilbert Map	Yes	Yes	Not Available	Live dynamic object tracking and prediction
FreMEn	N/A	Yes	Yes	Open Source	Binary data prediction (e.g. doors)
Hypertime	N/A	Yes	Yes	Open Source	Very versatile. Works with both binary and continuous observations

Figure 4.1: Summary of methods under consideration

4.4 Method Selection

After collecting a number of different potential methods for spatio-temporal world modeling, a project must then reduce the number of methods to a smaller, more manageable amount. The methodology for selecting methods to eliminate will vary between projects, but most often it is useful to identify properties of the desired method that are categorically missing from one or more other methods. In the case of ROPOD, it is important that the method be able to represent multiple dynamic objects, represent binary and non-binary states, and be either readily available for use in the form of a library or reasonable to implement in order to match the strict work plan timeline. Additionally, computational complexity, learning method, and offline vs online learning are important, they will not be the main focus of this section. The majority of these factors are best evaluated using expermential data and will be looked at in the experimental section.

With this in mind, the above methods will be evaluated one by one resulting in a pared down list of methods that can then be tested.

4.4.1 Occupancy Grids with Input-Output Hidden Markov Model (IOHMM)

IOHMM was originally designed for, and preforms best, when making predictions about the motion of an object from one cell to another in an occupancy grid. This is commonly used to track and predict human movement in a building or the flow of traffic at an intersection. Unfortunately, that poses an issue for this case study. Although motion prediction may be valuable for ROPOD in other scenarios, it is not applicable to this specific case study which places a focus on other areas of prediction. Another major issue this method has is its dependency on using an occupancy grid. Similar to the issue with motion prediction project, this specific case study uses a graph-like structure for path planning so although it may not be an issue in general, it is a problem for ROPOD. This method would also not be able to represent individual objects, providing yet another issue. Finally, with only parial availability it is clear that IOHMM is not a good fit for this case study.

4.4.2 Conditional Transition Maps (CTM)

CTM follows in the footsteps of IOHMM. Its dependency on occupancy grids as and the fact that it is best suited for motion prediction are major drawbacks. Additionally, with the lack of open source work availability, it is the easiest option to eliminate from this list.

4.4.3 Dynamic Multi-Map (DMM)

DMM, is an interesting method with respect to this case study. Despite its somewhat naive approach of averages, it does look suitable to the case study on paper. Importantly, it is map independent, allowing for the representation and prediction of both graph edges and individual objects. Additionally, it has no prediction restrictions on the type of behaviors or objects it can make predictions about. However, it has no known publicly available work which would be a drawback, but given its design simplicity, should be relatively easy to implement. DMM's lack of future or historical predictions is a bit of an inconvenience, but this can be overlooked as it is not strictly necessary for ROPOD. Given these reasons it will not be eliminated during the qualitative analysis.

4.4.4 Spatio-Temporal Hilbert Maps (STHM)

STHM is another very interesting method with respect to ROPOD. It's live dynamic object tracking and prediction, while useful for some cases of ROPOD, does not meet all of the evaluation requirements. Namely, this method focuses more on short-term motion predictions instead of long-term predictions that are needed for navigation with ROPOD. Much like CTM and STHM, is map dependent which is another drawback as the maps for ROPOD have already been chosen and defined. Unlike CTM and STHM, it does not have the motion-only prediction restriction

which does make it a bit more desirable. Unfortunately, the final eliminating factors for STHM are the lack of open source work availability. The lack of available work combined with the other issues listed mean that STHM is not suitable for this use case.

4.4.5 Frequency Map Enhancement (FreMEn)

FreMEn acts somewhat like a foil to STHM, at least with respect to its qualitative analysis and possible elimination. Its map independent which is a quick mark in its favor. Its very versatile and could represent both graph edges as well as individual objects in an environment. One major restriction and mark against it though is its inability to represent objects or behaviors that are non-binary, that is to say, objects or behaviors that have more than two states. Despite this restriction, the work is readily available which nudges it just on the other side of elimination. Time will have to be spent to work around this binary restriction, but ultimately, FreMEn may be a good solution to this case study. At the very least, it could fit well for representing if objects like doors and hallways are passible at a given time. Lastly, the ability to make historical or long-term future predictions is a minor, but appreciated plus.

4.4.6 Hypertime

Hypertime is perhaps the best-looking method on paper. Because it is an extension of FreMEn it's map independent and has historical and future predictions capabilities. Unlike FreMEn, and to its benefit, it does not have restrictions on the type of predictions it can make. This is a huge advantage. Finally, with preexisting work being readily available, Hypertime[1] is an excellent contender and will be tested in the experimental section.

[1]https://github.com/gestom

4.4.7 Results

Eliminated Methods

The following methods have been eliminated:

- Occupancy Grids with Input-Output Hidden Markov Model (IOHMM) - Modeling Spatial-Temporal Dynamics of Human Movements for Predicting Future Trajectories [20]
- Conditional Transition Maps (CTM) - Conditional Transition Maps: Learning Motion Patterns in Dynamic Environments [19]
- Spatio-Temporal Hilbert Maps (STHM) - Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments [12]

Methods Selected for Experimental Testing

The following methods will be tested further in the experimental section:

- Dynamic Multi-Map (DMM) - Dynamic Maps for Long-Term Operation of Mobile Service Robots [15]
- Frequency Map Enhancement (FreMEn) - FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments [18]
- Hypertime - Warped Hypertime Representations for Long-term Autonomy of Mobile Robots [17]

5

Experimental Setup

Having applied the comparison criteria developed in Section 3 to the ROPOD case study in Section 4 we have narrowed down the options for spatio-temporal world modeling in this particular case. However, a theoretical comparison offers only a partial solution. Given the focus ROPOD places on real-world environments it is critical that some operational tests be performed before selection of a method. Not only will these experiments serve as a guide for ROPOD, but they will also serve as a template for the comparison of future spatio-temporal world modeling techniques in other projects.

5.1 Environmental Representation

Given the complexity and size of the target environment for ROPOD, a large hospital, it is necessary to pare down features of the building until only the core components remain. The three dynamic environmental components being considered are doors, path planning with dynamic objects such as carts that are often strewn about the hallways, and elevators. To that effect, experiments have been designed to assess these key features. In the case of the doors and the elevator, the object under test, and its behavior, can be pulled out of the original complex environment. By isolating and simulating only the behavior of the object under test, the number of potential variables in the experiment is reduced. In the case of the path planning experiment, complete and total isolation from the operating environment is not possible. Instead, a graph has been designed to closely resemble an actual section of the

hospital along with the possible dynamic objects in that environment. Experiment specific details are covered in their respective sections.

5.2 Common Assumptions

In order to ensure only the desired component is being tested at any given time a set of assumptions are made.

- All robot components are working correctly (no internal faults)
- Other than the object under test (e.g. doors), all other objects in the environment are static
- All information other than the objects under test are perfectly known
- Observations made/provided by the training data are assumed to be ground-truth

5.3 Commonalities in Approach

Although different components will be under test, each experiment will be run in a similar manner.

The experimental setup is as follows:

- Training data consisting of observations made every 15 minutes over a simulated month will be provided to the models
- All data generation will be done using the same program and the specifics of how the data is generated will be discussed further below in the relevant experiment section
- Using the same generation procedure, a new month will be generated
- The models will be trained using the original month and then tested against both the original month and the new test month

- In the case of the doors and elevator, only the objects themselves will be modeled and the comparison will be how well the generated models are able to predict the ground truth
- When analyzing the more advanced hallway scenario, predictions about objects in the environment will be used to plan a path and this path will be compared with paths planned using the ground truth
- All of these paths will then be compared using the criteria described below.
- Computational resource usage will be tracked using the built-in Linux binary "time"[1]
- All experiments will be done on the same hardware and operating system
  - ASUS UX330UAK Laptop
  - i5-7200k 2.5GHz
  - 8GB DDR3
  - Arch Linux 4.19.4

5.4 Data Generation

Data generation is performed using a combination of built-in Python libraries and the NumPy[1] library. Each behavior to be modeled is broken down into a series of days which is then further broken down into a series of behaviors. Each behavior represents the likelihood of an object being in a given state between two times. Furthermore, each behavior has a starting state and an ending state which, if different, are swept through linearly, e.g., if a behavior is modeling the binary state of a door that starts at 100% open and then becomes 100% closed an hour later at the half hour mark the door would be defined as having a 50% likelihood of being open. Additionally, it is possible to specify the amount of Gaussian noise that is added on top of the behavior where the nominal state used is μ and σ is used to specify the magnitude of the noise. Months of data are thus generated by walking through these

[1]Specifically, "/usr/bin/time -f 'max_RSS=%MKB t_kernel=%S t_user=%U t_total=%E' $TEST was used"

[1]http://www.numpy.org/

behaviors in 15 minute increments for each day in the simulated month with each month assumed to have exactly 31 days.

As an example, if a door was open from noon until midnight every day and then closed from midnight until noon there would be two behaviors, one from midnight until noon and one from noon until right before midnight. In this case, the starting and ending values of each behavior would be the same and no noise would be introduced. This would create a sharp, well defined change at exactly noon and midnight every day. Further details and specifics about data generation will be discussed in their respective sections.

5.5 Model Parameters

Four models will be tested: DMM, FreMEn, Hypertime, and Gaussian. The first three models were selected using the comparison criteria in the previous section but, the keen reader will notice the inclusion of a fourth model. That model, the Gaussian Mixture Model is not expected to perform as well as the others. Despite this, it has been included for historical reasons as it has been used as a benchmark model in the past specifically in [2].

As for the other methods, Hypertime and DMM are of particular interest as they will be able to represent non-binary states. Hypertime will be using the expectation mean variant. FreMEn and Hypertime will both use an order of three as that has, on average, produced the best results as seen in [18]. During non-binary prediction Hypertime requires an upper limit for it estimates. In the case of the elevator experiment, an upper limit of 30 (seconds) has been given. This will be expanded upon in Section 6. The DMM model will contain three sub-models. The models will have a .75, .25, and .05 replacement rate. All DMM maps/objects will have a size of 20. This is to closely match the approach in [15]. The three internal models will be averaged to produce a prediction.

It is important to note that unlike the other three approaches, DMM does not inherently support the ability to predict events arbitrarily in the future. Because

it is limited to a single immediate prediction and future predictions will done by simply projecting historical predictions forward on a monthly basis. That is to say, whatever a given DMM's prediction was on the third day of the first month will be considered the same for the third of the next month.

The DMM implementation was implemented using Python 3 while Gaussian Mixture Models, FreMEn, and Hypertime were all written in C++. The DMM implementation was written by the author of this paper while the other three are available on Tom Krajnik's Github[1] page. Finally, the non-binary version of Hypertime, although not yet freely available on Github, can be obtained upon request as mentioned in the binary-only repository.

5.6 Comparison Criteria

In order to evaluate the intricacies of the selected models, a wide variety of data points have been selected for comparison. A focus has been placed on collecting data relevant to accuracy and scaleability of the modeling technique given the eventual scope of the ROPOD project.

- Accuracy to Ground Truth
- Accuracy to Historical Recreations
- Training and Planning Run-time
- Training and Planning Memory Consumption

5.7 Doors as Dynamic Objects

5.7.1 Experimental Motivation

As is the case in many places of employment, many areas of a building may not be accessible to the public, and by extension the robots, outside of work hours. This

[1] https://github.com/gestom

could come in the form of a given hallway between two areas being locked when public access ends or workers leave for the day. In another case, it could be as simple as someone preferring to having a door to a hallway shut during a particularly loud time of the day.

Regardless of why, it is certain that the states of doors are often dynamic and exhibit both periodic and long-term changes. In the ideal case, a robot would learn when certain doors are closed and be able to plan accordingly. While making an accurate prediction can save time, making an inaccurate predication can be extremely costly. An inaccurate prediction would force a robot to not only backtrack, but also recalculate the path required to get to a target. Additionally, it may not be possible to make deliveries at all times. In the worst case, bad pathing can lead to a robot becoming locked in an environment and making it unable to return to its base thus eventually running out of power and requiring human intervention. For these reasons and more, the door experiment highlights the benefits and importance of selecting the correct spatio-temporal world model.

5.7.2 Experimental Details

In order to keep this test as simple as possible, only the door object itself has been modeled. For this example the doors could belong to a room where a package must delivered or perhaps a hallway that could be used as a shortcut. Each model will be directly or indirectly tasked with predicting the state of a given door. Figure 5.1 displays an example of three doors in ward 24 of the hospital that may display the dynamic behavior previously mentioned.

When evaluating the predictions made by the models, a simple 50% confidence value will be used. That is to say, if a model predicts that the door will be 50% or more likely to be open, the robot will attempt to deliver the package.

It is clear that this 50% cutoff does not take into account the penalties of making a wrong prediction, but this particular experiment is designed only to investigate

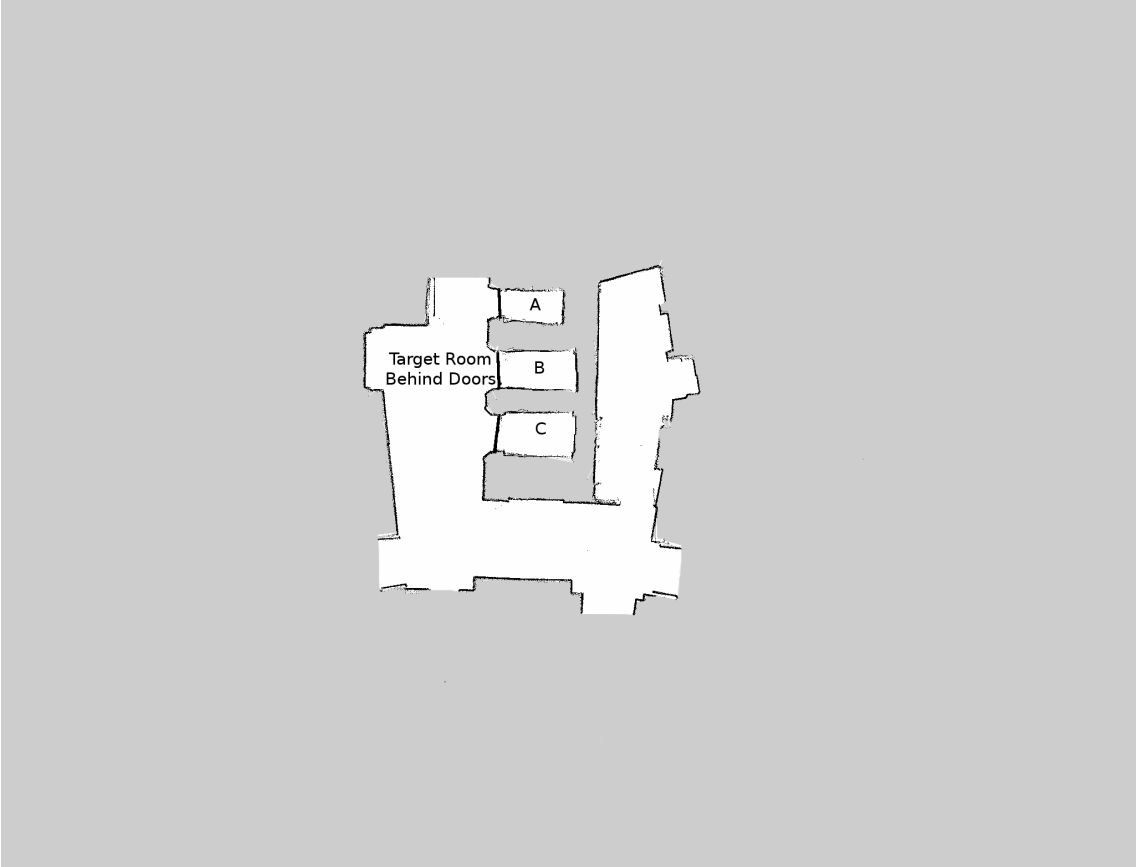

Figure 5.1: Multiple rooms behind doors in ward 24.

the accuracy of the prediction. The hallway experiment discussed below will delve deeper into the ramifications of inaccurate predictions. The result of a incorrect prediction being carried out is undoubtedly valuable, but is outside the scope of this specific experiment.

5.7.3 Data Generation

As mentioned above, both the training and the test data for all models was generated using the same program. The specifics for each door are as follows.

Door A

Door A models a door that is highly influenced by the daily schedule of a 9:00 to 17:00 job and has a high rate of periodic behavior with a small amount of noise. There are 5 behaviors which model a standard work day. The time from midnight until 9:00 has the door in a normally closed state. From 9:00 until 12:00 the door is likely open. 12:00 until 13:00 is considered lunch time and thus the door is likely closed. Work resumes at 13:00 until 17:00 and thus the door is likely open. From 17:00 until midnight the door again remains in a likely closed state. Finally, the door is always closed on weekends with a 100% probability and no noise.

When the door is in a likely open state it has 70% chance of being open. Likewise, when in a likely closed state it has a 70% chance of being closed. Finally, when noise is introduced during the weekday it uses a standard deviation of 0.1 where μ is either 0.3 or 0.7 and 0.5 is the cutoff for being open or closed. Figure 5.2 shows an example of what a month of this behavior looks like.

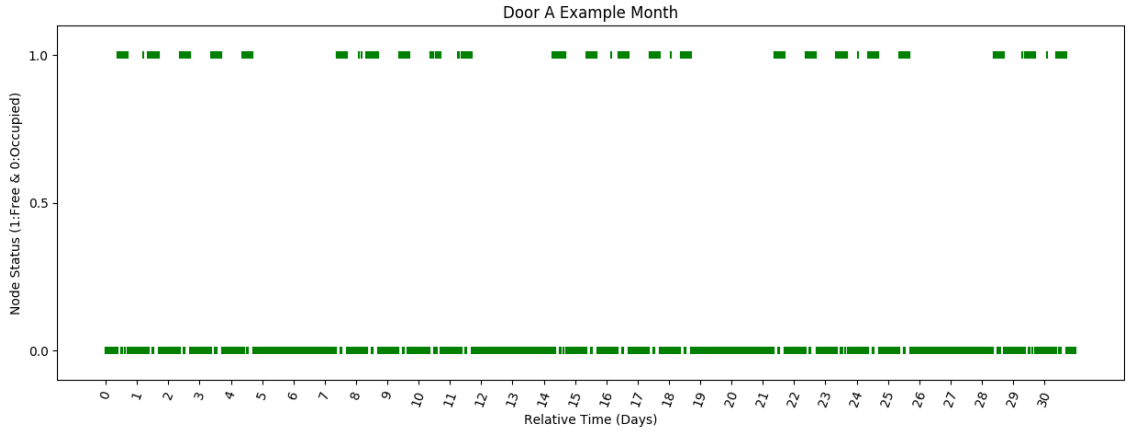

Figure 5.2: An example month of Door A behavior

Door B

Door B, in a similar manner to door A, also tries to encapsulate the periodic behaviors of a work day but with slightly more noise. Door B is always open from midnight

until 9:00. During the work day the door is constant state of flux, being open and closed at random. This is modeled using a μ of 0.5 and a standard deviation of 0.1. This flux ends when work hours end at 17:00 and the reverts back to remaining open until the next work day starts at 9:00. To add additional periodic complexity, such as a weekly meeting or delivery, every third weekday the door will be closed the entire day. This trend is continued across month boundaries meaning if the doors were closed on the day before the last day of the first month it will again be closed on the second day of the next month. Finally, on weekends the door will always be open with 100% certainty and no noise. An example of this behavior is found below in figure 5.3.

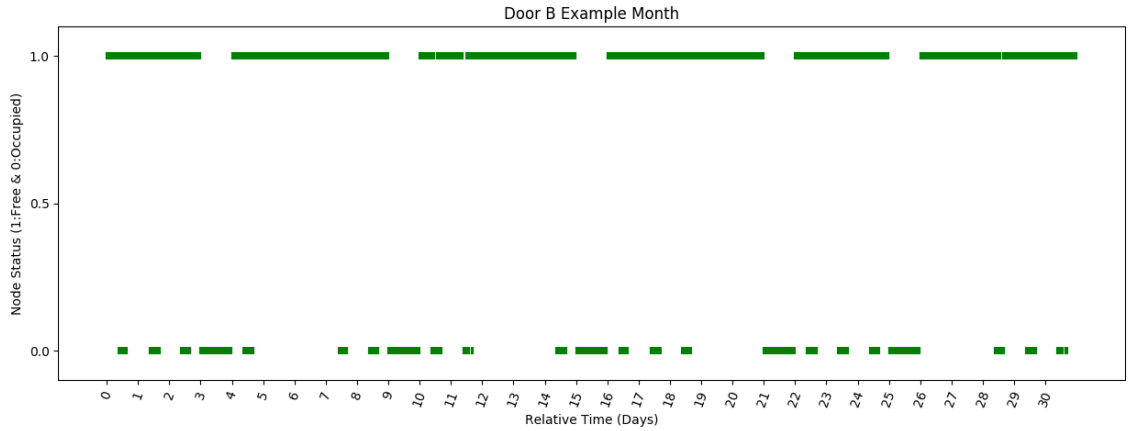

Figure 5.3: An example month of Door B behavior

Door C

Door C does not attempt to model any periodic behavior but instead test long-term change in an environment. A simple example of the long-term non-periodic change is construction. The door has always been open in the past, but leads to a wing of the building that is now either being remodeled or has been removed. In the test data, this behavior is achieved by having the door be open for the first three weeks and then being closed for the rest of the training month and through the test month. In this model no noise was introduced. Figure 5.4 shows an example of this behavior.

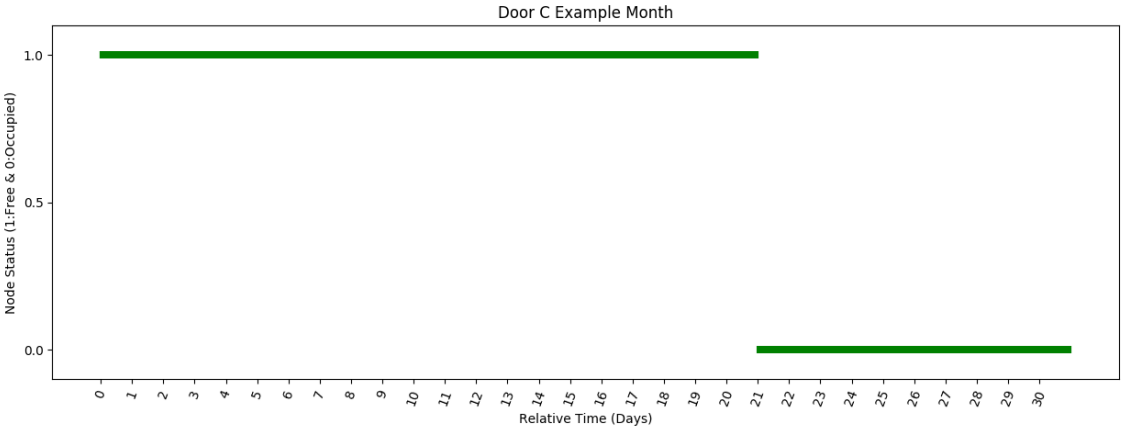

Figure 5.4: An example month of Door C behavior

5.8 Path Planning in Congested Hallways

5.8.1 Experimental Motivation

Individual object representation and performance is vital for initial evaluation of an algorithm, but more expansive tests must be completed to build a complete picture of an algorithms real-world performance. It is for this reason that multiple dynamic objects will be used to model an environment that closely replicates ROPOD's target environment. Not only does this serve as an excellent test case for ROPOD, but it also serves as an initial evaluation into the scalability of the proposed method.

When planning paths with multiple dynamic objects it is critical that not just a few, but all of the objects are correctly predicted. If any of the objects' states are predicted incorrectly, a robot may need to reroute dynamically or it may not be able to arrive at is destination thus wasting time and resources.

5.8.2 Experimental Details

The specific area under test in this experiment is a representation of the basement area of the hospital ROPOD will be deployed in. Figure 5.5 is a cleaned up version

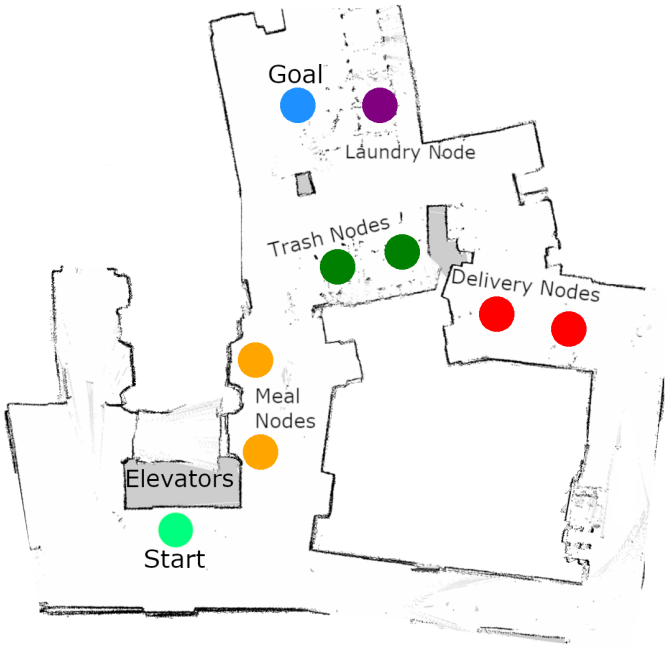

Figure 5.5: Congested Basement with marked nodes

of ROPOD-generated map. Areas have been marked to highlight where common dynamic obstacles occur. Figure 5.6 shows a graph representation of this same area, simplified for path planning. Black nodes are static and not traversable. The start

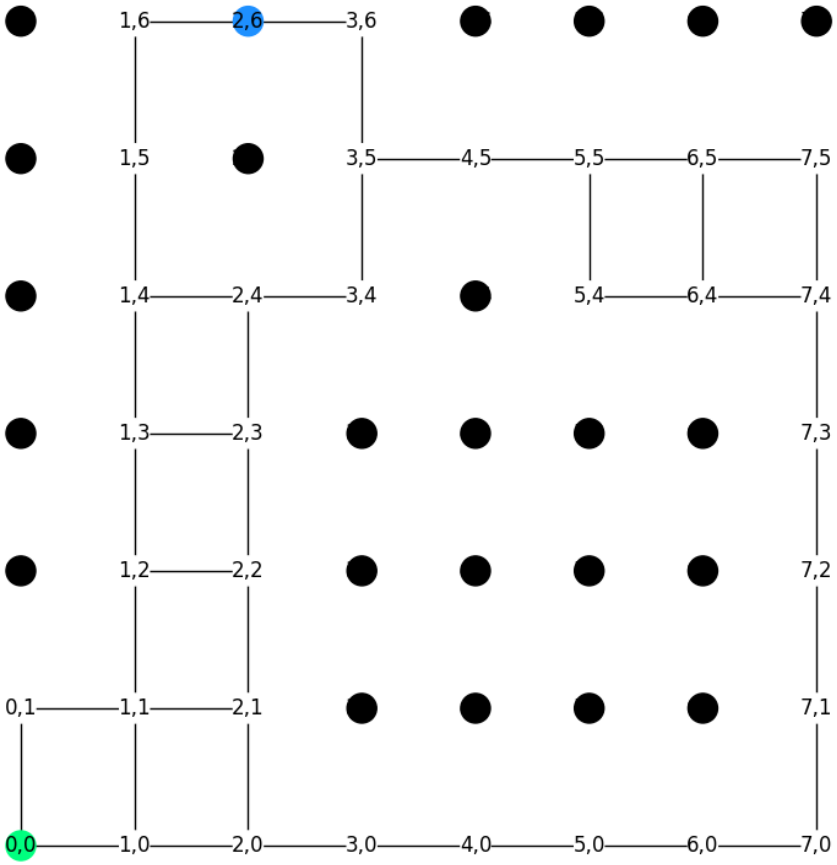

Figure 5.6: Graph representation of the hospital basement with no obstacles

and goal nodes have been marked and all connected nodes are traversable. Finally, figure 5.7 shows the same area but with all dynamic obstacles present. Notice how when all obstacles are present it is no longer possible to get from the start to the goal. The obstacles and their frequencies are as follows:

- Meals - three times a day
- Laundry - once per day
- Delivery - twice per week
- Trash - once every three days

The edges in the graphs are the possible routes the robot is able to take. Node 0,0 represents the start state which is roughly where the robot would be after taking the elevator to this floor. Node 2,6 is the goal state and represents an entrance into either a storage area or another hallway.

5.8.3 Data Generation

The data for these tests was generated using the same program as the doors and in the same manner. The graph generation and path planning was done using NetworkX [1] a free and open source tool that allows for, amongst other functionality, the creation and subsequent path planning on graphs.

The specifics for each door are as follows.

Meals - Nodes 1,2 & 1,2

Meals are a periodic and consistent behavior in the hospital. They happen three times a day regardless of the day of the week. Meals and their associated materials are staged in the hallway before being moved to another area or delivered. For ease of simulation, carts begin to pile up at node 1,2 at 4:00 and continue to build up until overflowing into node 1,3 at 6:00. These two nodes continue to be occupied until 8:00 when all carts are sent up for delivery, at which point those nodes are again free for traversal. The process repeats again starting at 12:00 and 16:00 using the same time steps.

[1] More information can be found on online at https://networkx.github.io/

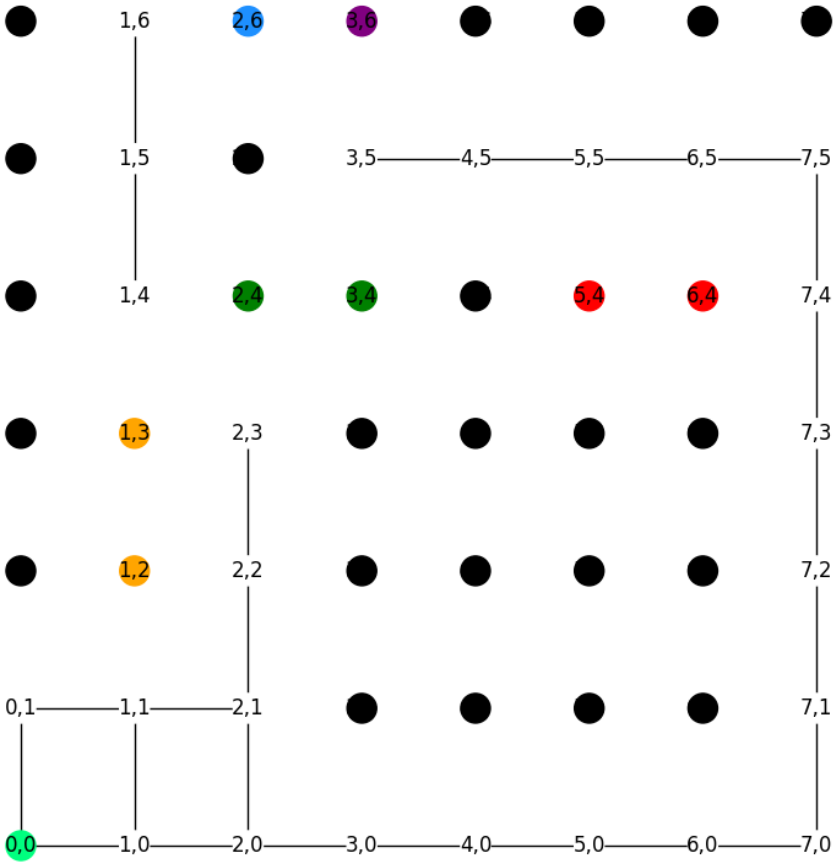

Figure 5.7: Graph representation of the hospital basement with all obstacles modeled

Laundry - Node 3,6

Laundry, similar to meals, is assumed to build up periodically and daily causing node 3,6 to be blocked. Laundry begins to build up slightly before midnight and begins to cause blockage at 00:00 until 12:00, at which point enough laundry has been processed that the node is again traversable.

Deliveries - Node 5,4 & 6,4

Deliveries are assumed to happen twice a week, once on Monday and then again on Friday. For the sake of simulation, deliveries block nodes 5,4 and 6,4 the entire day of the delivery, i.e. from 00:00 until 23:59.

Trash - Node 2,4 & 3,4

Trash or other nondescript objects are periodically stored in the hallway. For the sake of simulation, trash begins to build up at node 3,4 and then expands into node 2,4. On the first day of the first month both nodes will be free for traversal. After the first day, at precisely 00:01, the first node is said to be blocked. After another day, the trash has accumulated enough to block the second node. This cyclic behavior continues when the second test month is generated. In summary, the first day both nodes are free, the second day the first node is occupied, the third both are occupied, and then on the fourth day they are back to both being free for traversal.

Areas of Particular Note

As visible in figure 5.7 it is not always possible to reach the goal from the starting node. Additionally, certain obstacles block paths that might normally be considered first or optimal. An example is the obstacles caused by the meal nodes. They obstruct the most direct path to the goal upon leaving the elevator. Choke points are also a concern. These happen when an obstacle or obstacles completely cut off a given path. Examples of this are when both laundry and the first trash node are present or when both trash nodes and both meal nodes are present.

5.9 Busy Elevators

5.9.1 Experimental Motivation

Historically, many efforts in the field of spatio-temporal world modeling have been focused on modeling or classifying behavior into one of two binary states. Although this works well for representing the traverseability of an environment or detecting the presence of an object, it does not translate well into the tracking and

predicting the behavior and state of many real-world objects. The ability to track and predict non-binary behavior is particularly important when a robot shares an environment with humans. These environments are dynamic and contain many non-binary variables such as the speed at which hallway traffic flows.

With regard to ROPOD, an excellent example of this non-binary behavior is the necessity to use elevators to move goods about the hospital. Specifically, the time between calling an elevator and the actual arrival of the elevator may often appear random, but when observed throughout time it is possible to observe patterns in the wait time. In order to better optimize which elevator should be called or when a delivery should be made, it is important to have accurate models for predicting this behavior. Additionally, although this information alone is useful, this concept can be applied to a multitude of areas within a dynamic human environment.

5.9.2 Adaption of Non-Binary Models

Two of the models being tested, Hypertime and DMM, nativity support classification of non-binary data. Additionally, although FreMEn and Gaussian Mixture Models do not natively support the prediction of non-binary states, they have been modified in order to produce results for analysis. Before training either of the models, the datasets are ordered and split in half at the median. The median value is then used for classification. If an observed value is below the median it is considered a 0 and considered a 1 if above the median. In this way, a continuous dataset can be broken into two states. Predictions are then made as normal. In order to convert back to a non-binary value, each half of the split data set is averaged to produce an average value for all values classified as 0 and all values classified as 1. These averages are then used for predictions.

For example, if the original dataset contained six observations, [1, 2, 3, 4, 5, 6] the dataset would be split into [1, 2, 3] and [4, 5, 6]. Any prediction greater than or equal to 4 would be a 1, and lower a 0. All 0's during prediction would then be the average of the first dataset, e.g. 2 and all 1's would result in a prediction of 5.

5.9.3 Data Generation

As mentioned in the two previous sections, both the training and the test data for all models was generated using the same program. However, due to the nature of the non-binary data slight changes were made in order to represent any set of arbitrary real numbers. Behaviors no longer represent the likelihood of an object being in one of two states, but instead, in this specific case, represent the average time one must wait after calling an elevator. There are still linear transitions between behaviors as well as a variable amount Gaussian noise being applied on top of the nominal state (i.e., noise applied on top of the average wait time). An example month of data is visible in figure 5.8.

Average Daily Model

In this particular simulation, everyday was treated as the same type of day for the month long simulations. Each day starts and ends with almost no wait time to simulate how infrequent the elevators are used during those times. Peak elevator use time, and thus peak delays, are said to occur around times when people would being arriving or leaving work or lunch. The specifics of the behaviors and their parameters are visible in Table 5.1.

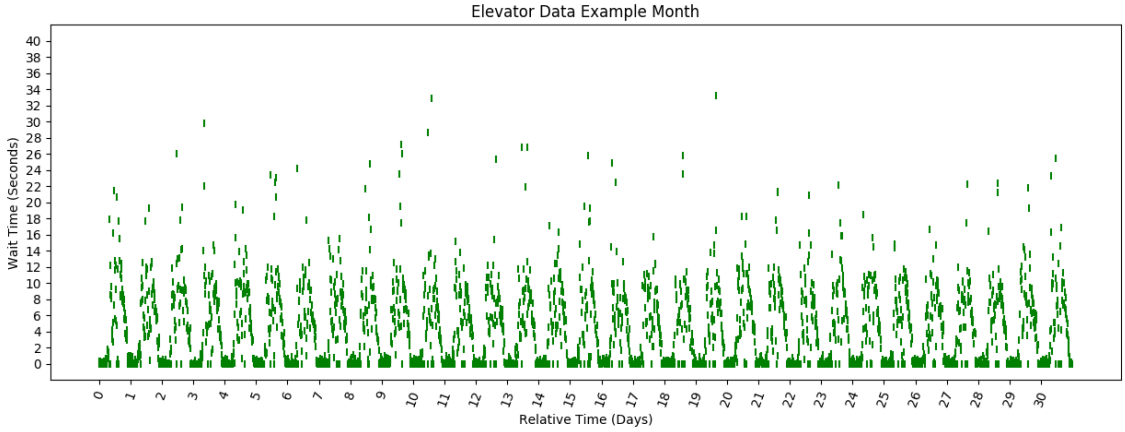

Figure 5.8: The training data for the first elevator month

Behavior Name	End Time	μ in Seconds	σ in Seconds	Max Wait
Morning Lull	6:00	0	0.4	2
Morning Early Arrivals	7:30	0	0.5	N/A
Morning Pre Rush	8:30	2	2	N/A
Morning Work	10:00	9	1	N/A
Early Lunch	11:00	4	1	N/A
Lunch Pre Rush	11:30	4	2	N/A
Lunch Rush	12:30	9	2	N/A
Midday Work	13:30	9	1	N/A
Evening Pre Rush	16:00	4	2	N/A
Evening Rush	17:30	9	2	N/A
Day Wind Down	22:00	9	0.5	N/A
Day End	24:00	0	0.4	2

μ - Nominal Wait

σ - Noise

Table 5.1: Behaviors used to describe daily wait times for the elevator

5.9.4 High Resolution Data

Finally, an additional dataset has been included for comparison. During normal data generation, observations are assumed to occur consistently every 15 minutes. In the additional "high resolution" dataset, observations happen every 1 minute. The inclusion of this additional dataset aims to outline how the models perform with larger sets of data while keeping the actual underlying data similar to the standard resolution dataset.

6

Experimental Results

6.1 Doored Areas

6.1.1 Door A

Door A, is an excellent starting point and litmus test for various spatio-temporal world modeling techniques due to its periodic, humanlike, and slightly random nature. Door A's results establish a trend that can be seen throughout the rest of the results. This is particularly visible in figure 6.3 where the accuracy of each model is evaluated over time. Table 6.1 also establishes a trend with respect to accuracy and computational resource usage that is mirrored in the majority of experiments.

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	84.11%	77.80%	92.51%	97.66%
Prediction Accuracy	77.15%	79.43%	87.08%	88.87%
Computation Time (Milliseconds)	610	50	70	5630
Memory Usage (KB)	31036	34968	34656	37192

Table 6.1: Door A Data Overview

Due to the periodic nature of the data, the techniques based off of extracting periodicities (i.e. FreMEn and Hypertime) do an excellent job of recreating past events as well as predicting future events. The prediction of past and future events

can be seen in figure 6.1 and 6.2 respectively. HyperTime's slightly better prediction ability is most likely due to the extra steps taken to wrap the data back in on itself temporally. However, this additional training and accuracy comes at a great cost. Hypertime manages only a small performance increase of around one percent over FreMEn with the total training and prediction time taking on the order of two magnitudes longer when compared to either Gaussian or FreMEn. Finally, it appears that the Gaussian model completely failed to predict any open state. This is likely due to the sparseness of positive, or open states resulting in the model oscillating between 0 and just under 0.5 thus never reaching the open threshold.

6.1.2 Door B

Door B demonstrates a fact the will be come increasingly clear with future experiments. Methods that use a modified version of the Fourier transform as described in [18] require a certain threshold of frequency to be met in order to accurately make predictions. In fact, its interesting to note that the very thing that allows these methods perform so well with periodic behavior causes issues within datasets with non periodic behavior or datasets with minimal periods of periodic data.

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	85.71%	59.81%	75.20%	71.55%
Prediction Accuracy	69.24%	62.17%	76.95%	75.78%
Computation Time (Milliseconds)	600	60	80	1440
Memory Usage (KB)	31036	34644	34892	37692

Table 6.2: Door B Data Overview

DMM, relying almost solely on averages, does surprisingly well in this experiment, beating all other models in both historical and prediction accuracy visible in figure 6.4 and 6.5. It is, however, important to note the flaws in DMM's long-term prediction. Due to the fact that future predictions are not directly possible using DMM and thus historical data is used, it is clear that, while DMM performed well overall historically,

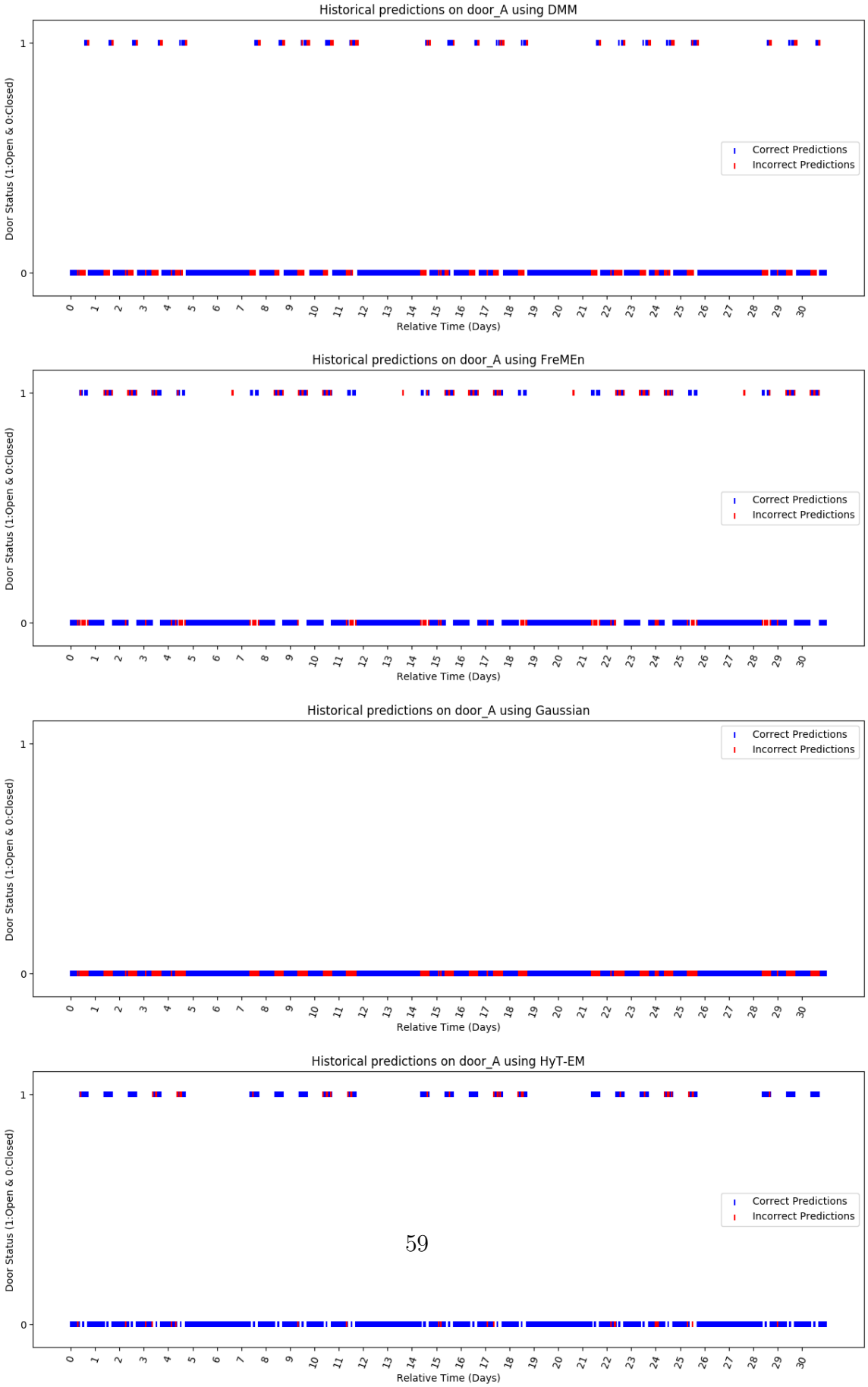

Figure 6.1: Historical Recreations - Door A

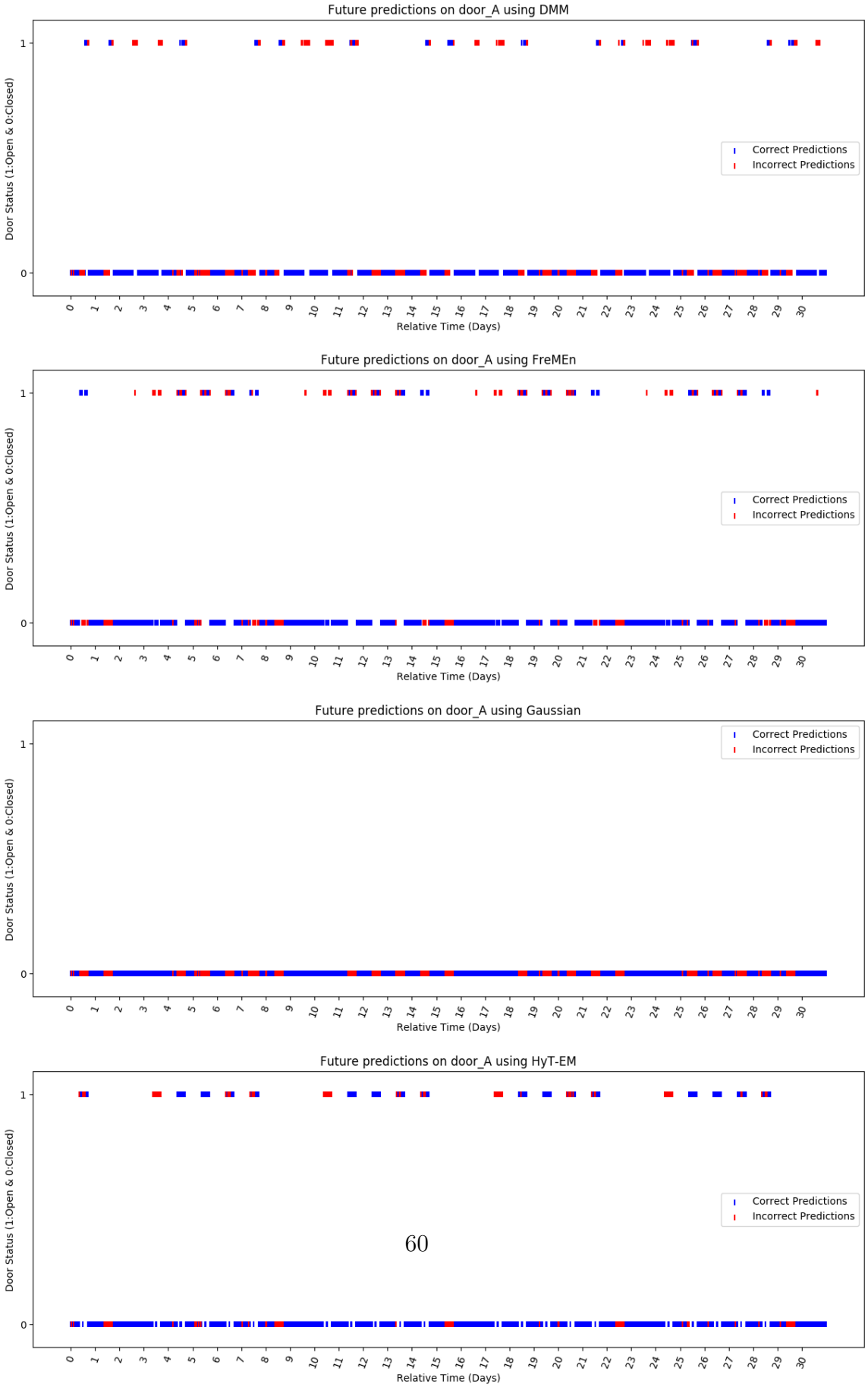

Figure 6.2: Future Predictions - Door A

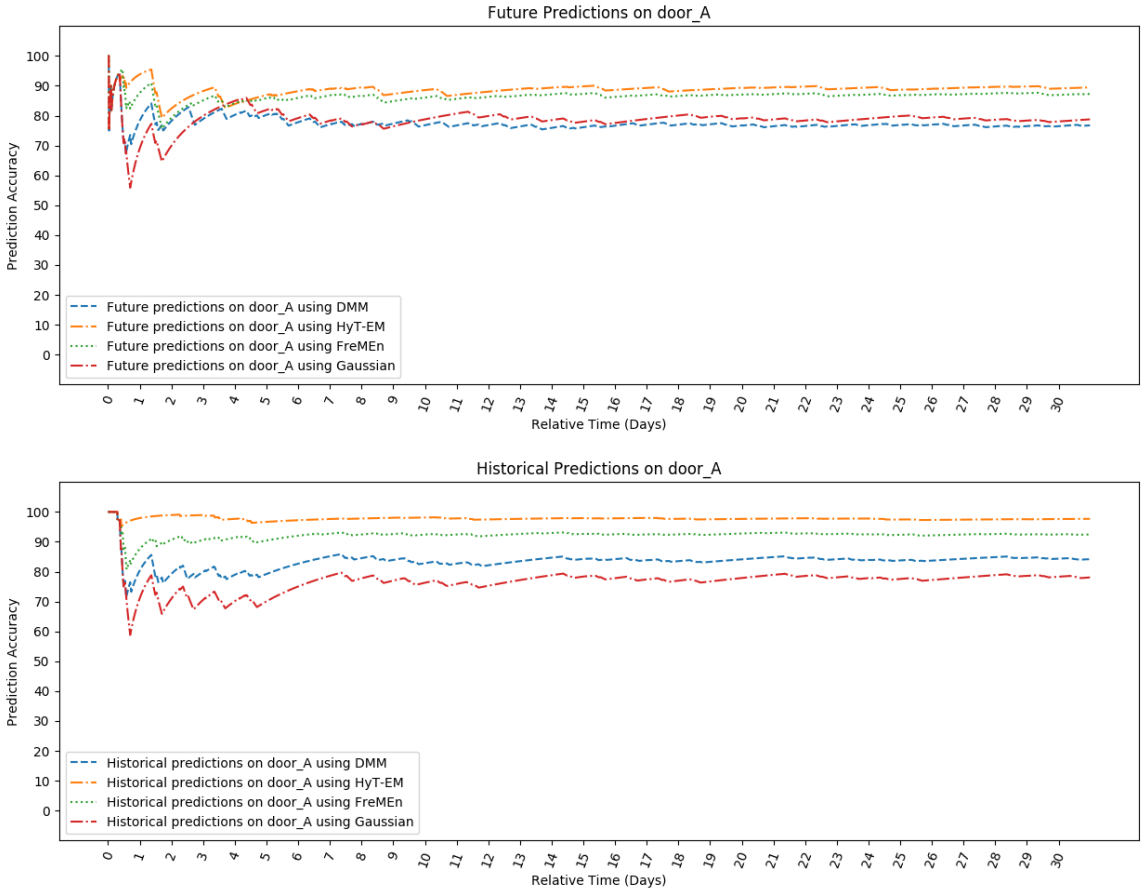

Figure 6.3: Model Accuracy Over Time - Door A

it is not without problems in future prediction. Of particular interest is DMM's failure to continue to accurately predict a periodic behavior that does not occur on month boundaries. Since the behavior that happens every three days does not happen on the same day between the two months DMM incorrectly predicts its occurrence as happening on the same days each month.

In terms of resource usage, a similar trend to door A is observed as visible in 6.2. Gaussian and FreMEn predictions take under 100 milliseconds while DMM takes around an order of magnitude longer, and Hypertime yet another order of magnitude beyond that. Finally, again, similar to door A, all approaches use roughly the same amount of memory for training and predicting at around 30 megabytes and within about 10 megabytes of one another.

6.1.3 Door C

With door A exemplifying the occasionally periodic and somewhat noisy behaviour in the real-world, door C serves as almost the exact opposite, modeling a one time, long-term change. Perhaps expectedly, the results in terms of prediction accuracy are almost completely opposite to that of door A with DMM having the best performance both historically and with future predictions.

Unfortunately, despite DMM's better performance by the numbers in table 6.3 looking at the resulting graphs in figures 6.7 and 6.8 show a somewhat initially disappointing result. As discussed in the door B experiment, DMM, merely uses its historical predictions again for future predictions. This is clearly visible by the prediction of door C being open for the first three weeks in the future predictions. However, DMM was not designed for long-term future predictions and is instead meant to be used "live". When this is accounted for, DMM's performance is once again impressive. In fact, as mentioned in the previous section the historical predictions made by DMM can be viewed as it's "live" predictions and thus it would be expected

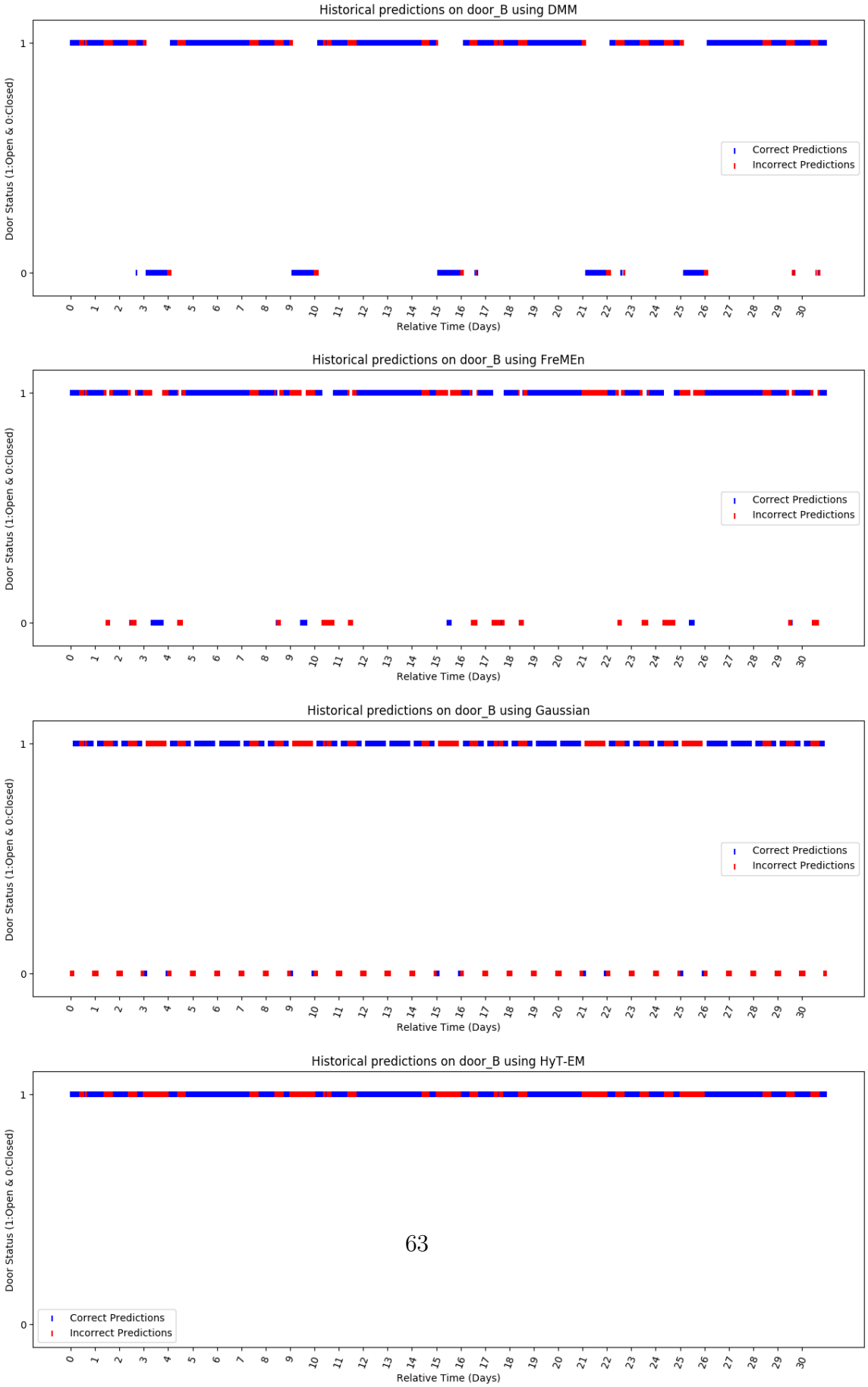

Figure 6.4: Historical Recreations - Door B

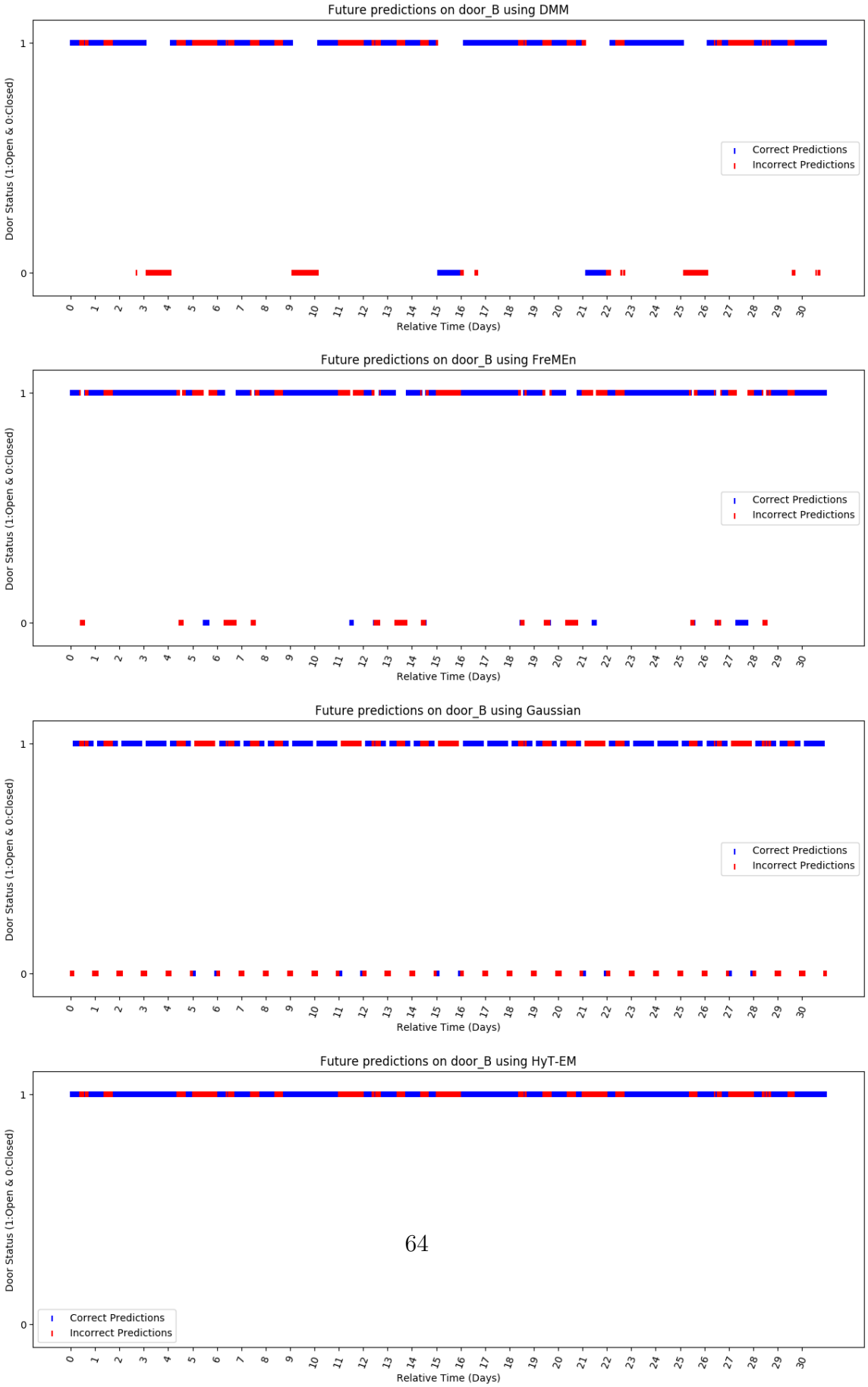

Figure 6.5: Future Predictions - Door B

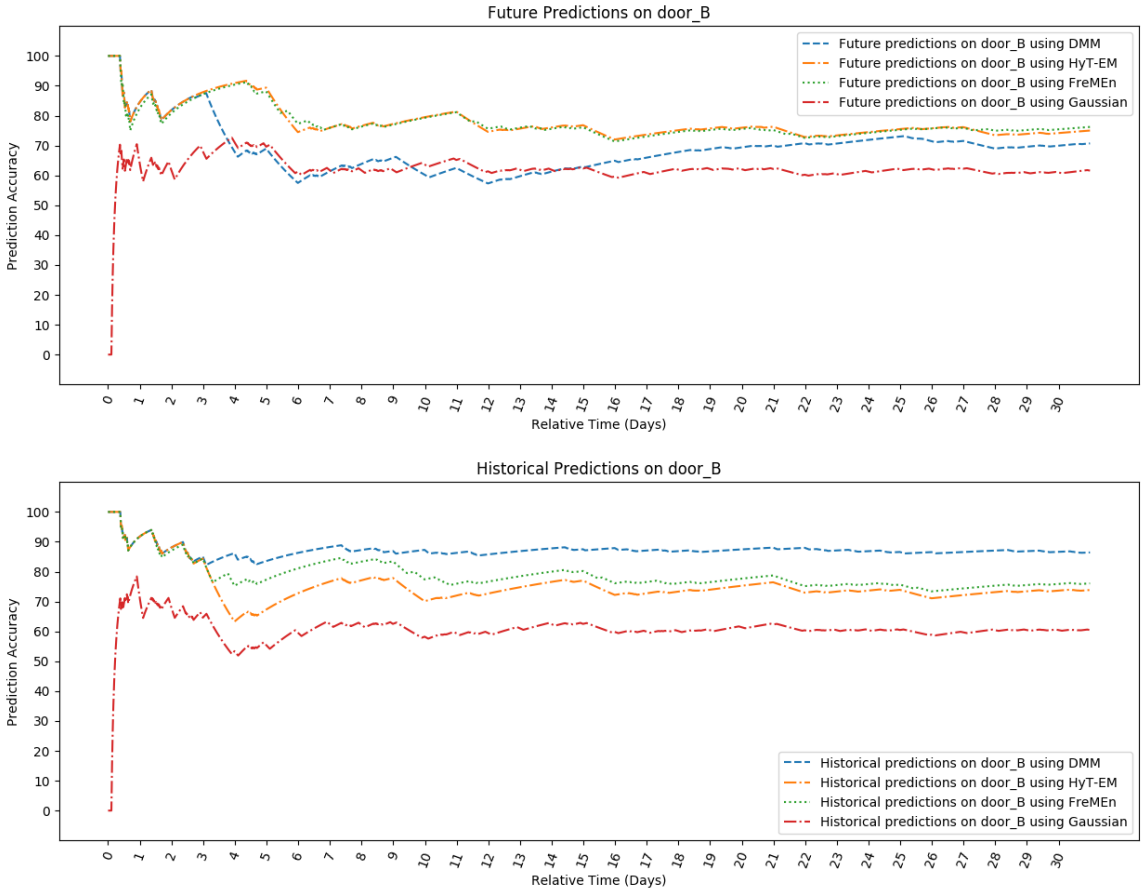

Figure 6.6: Model Accuracy Over Time - Door B

that in the real-world it would continue to predict the door as being closed as long as long-term future predictions are not requested. Thus it is likely that DMM's performance would be closer to 100% for future actives for as long as the door remained shut.

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	99.56%	62.75%	67.74%	67.74%
Prediction Accuracy	31.82%	14.58%	00.00%	00.00%
Computation Time (Milliseconds)	570	60	70	530
Memory Usage (KB)	31224	35004	34976	37208

Table 6.3: Door C Data Overview

As alluded to above in door B's experiment, the lack of periodic data has caused both FreMEn and Hypertime to take the easiest prediction for the training data and predict that the door will always be open. Unfortunately, this causes the future predictions, seen in figure 6.8 to be entirely inaccurate. The only possible redemption for the Fourier based approaches on this long-term change is that the prediction would eventually flip to always being closed, but only after the total number of observations of the door being closed surpassed that of the door being open. In this case, that would take a total of just over 6 weeks if training was being done every night.

The resource usage statistics in table 6.3 do interestingly break slightly from the norm. It appears that due to the simplistic predictions of the Fourier approaches, Hypertime has shaved off an order of magnitude in computation time. This is believed to be because the prediction model created by Hypertime, after the initial naive approach, has a larger error than the first and thus it immediately quits and stops attempting to improve the model. This is irrelevant however, seeing as the predictions are completely inaccurate so any gain in computational time is meaningless as seen in figure 6.9

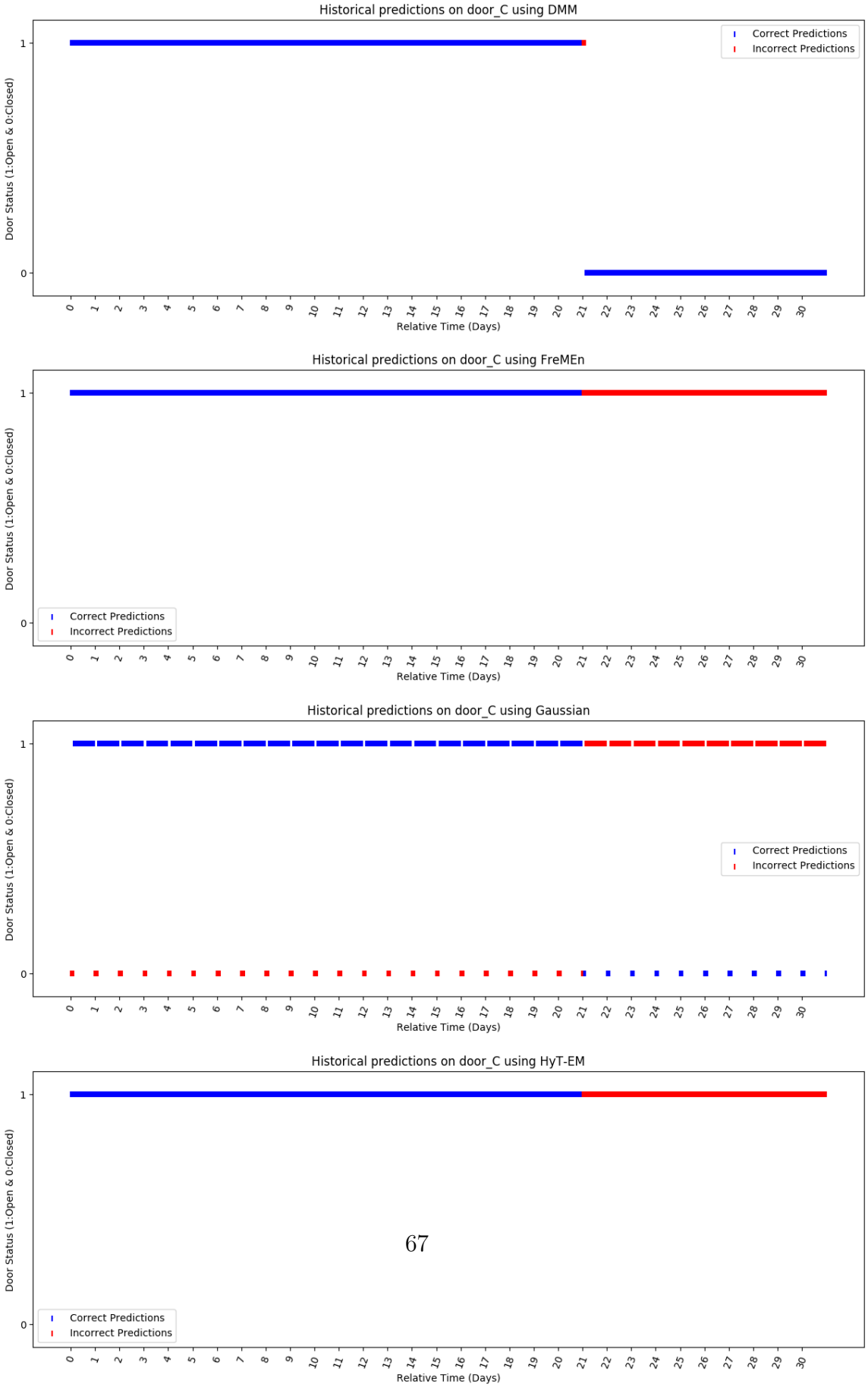

Figure 6.7: Historical Recreations - Door C

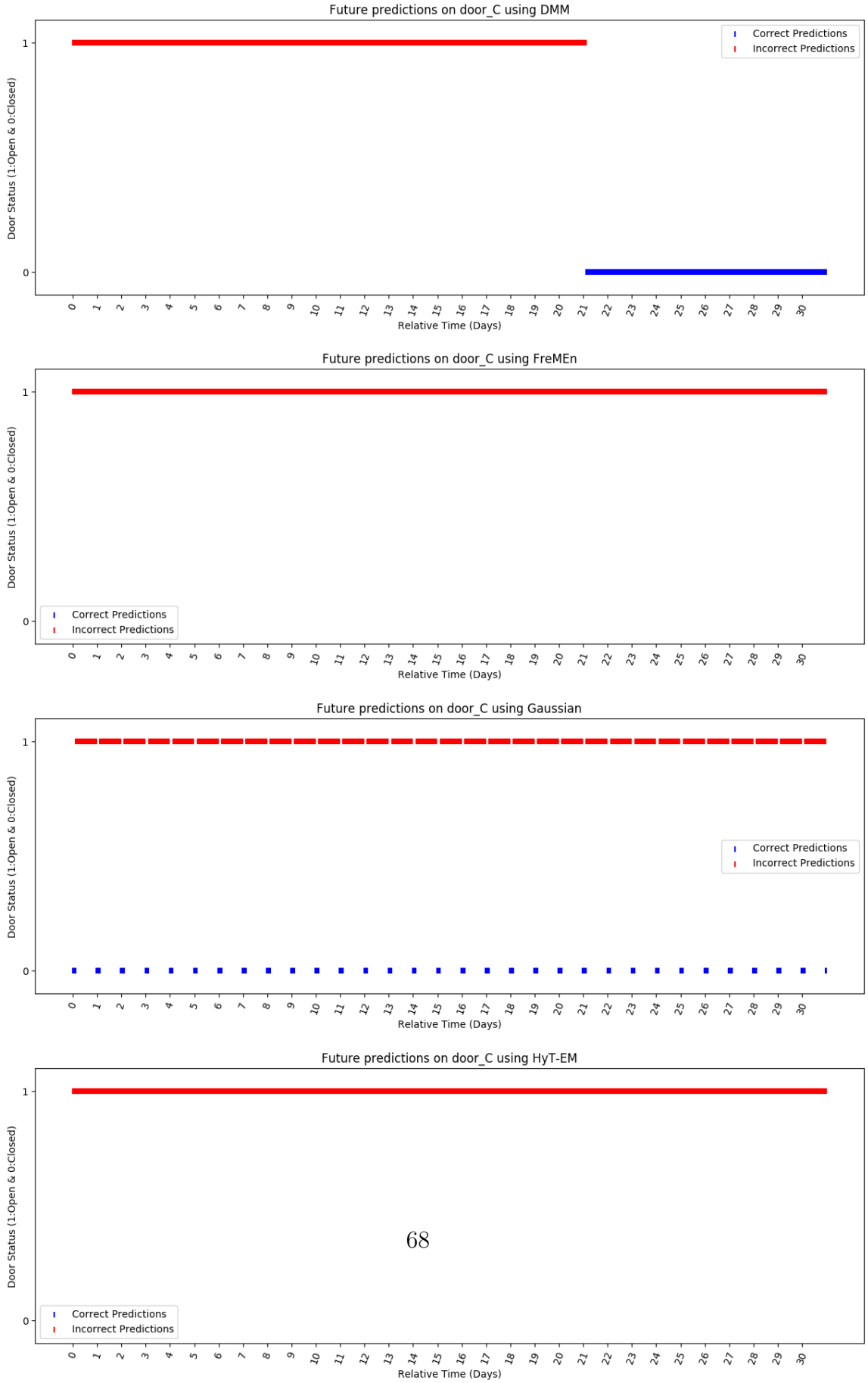

Figure 6.8: Future Predictions - Door C

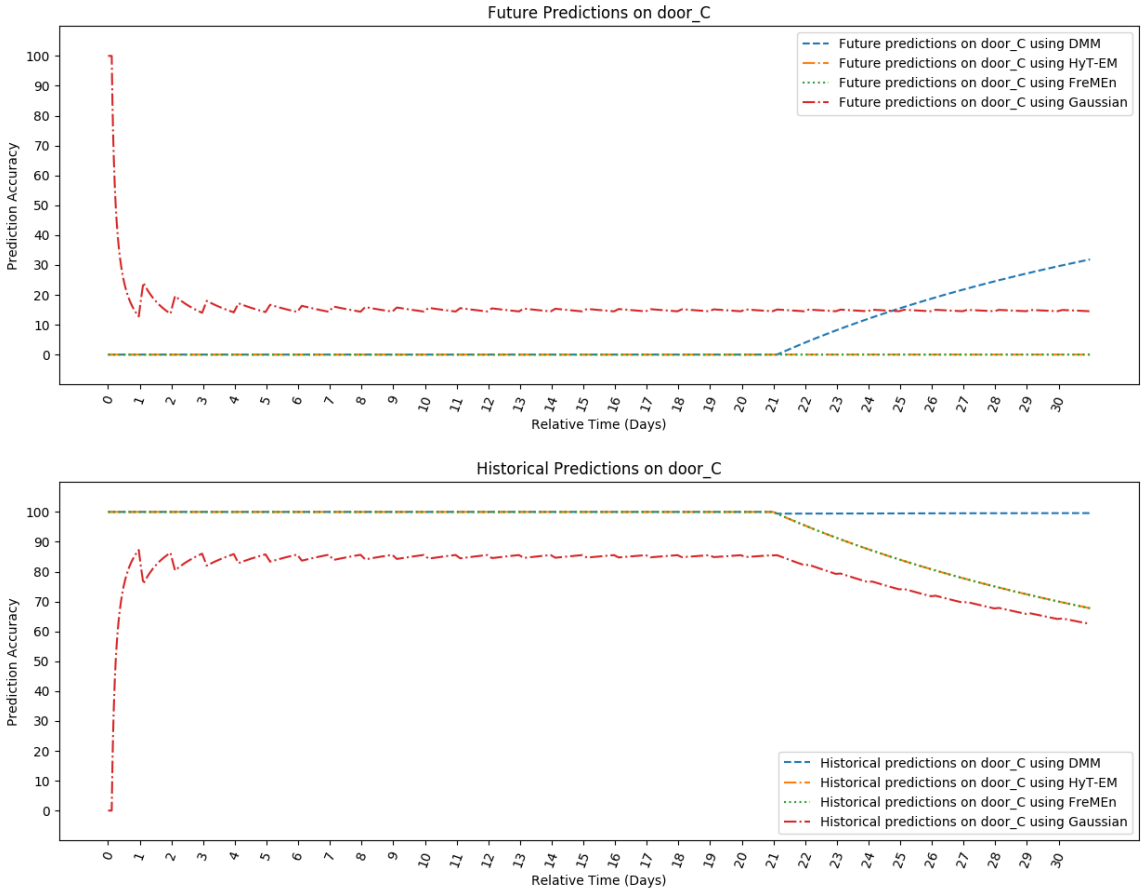

Figure 6.9: Model Accuracy Over Time - Door C

6.2 Congested Hallways

Having already given an in-depth look at the individual predictions in the previous section, this section will focus on metainformation that can be abstracted and an evaluation of performance with respect to path planning which is the intended primary use case. To that effect, graphs for every single node are present in the appendix but have not been included in this section.

6.2.1 Prediction Accuracy Versus Behavior Frequency

With a moderately large dataset that has varying frequencies such as the one used for this experiment, it is possible to observe a relationship between the accuracy of a given spatio-temporal modeling technique and the period of the behavior attempting to be modeled. In general, the data suggest that behaviors with a higher frequency are better modeled using Fourier methods like FreMEn and Hypertime. This is most obvious when looking at behaviors that are repeated multiple times a day such as the laundry or meal nodes as seen in tables 6.4, 6.5, and 6.6. In these experiments, the Fourier methods achieved an impressive 100% accuracy. One caveat to these impressive results is the lack of noise present in the data used for the hallway test. Noise in the data was explicitly left out of this particular experiment to minimize the variables present. The presence and effect of noise in data is, however, analyzed in both the door and elevator experiments.

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	78.93%	26.04%	100.00%	100.00%
Prediction Accuracy	78.93%	26.04%	100.00%	100.00%
Computation Time (Milliseconds)	610	60	70	1860
Memory Usage (KB)	30984	35160	35172	37632

Table 6.4: Hallway Laundry Section

When observing the behavior that occurs with the least frequency, the trash nodes, table 6.8 and 6.8, the opposite trend is observed. Fourier methods no longer are able to accurately predict behaviors and their predictions drop to around 66% accuracy. This is not believed to be a problem with predicting or modeling behaviors

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	61.63%	52.08%	100.00%	100.00%
Prediction Accuracy	61.63%	52.08%	100.00%	100.00%
Computation Time (Milliseconds)	620	60	70	880
Memory Usage (KB)	30896	35524	35600	38288

Table 6.5: Hallway Meal Section 0

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	74.93%	65.62%	100.00%	100.00%
Prediction Accuracy	74.93%	65.62%	100.00%	100.00%
Computation Time (Milliseconds)	600	60	70	990
Memory Usage (KB)	31384	35388	35588	37608

Table 6.6: Hallway Meal Section 1

with a lower frequency, but rather a consequence of the ratio between frequency and length of observation. It is postulated that with a larger dataset, achieved by increasing the time of observation while maintaining the density of observations, the predictions would improve.

While Fourier methods appear directly correlated with frequency, DMM appears to have an inverse relationship with frequency. In the meal and laundry nodes DMM has a prediction accuracy between around 60% to 80%. Furthermore, the lowest prediction accuracy, 61.63% is found on the meal node that happens to have the highest number of changes per day. It is likely that the poor behavior of DMM on behaviors with high frequency is due to how the averages are stored and computed. DMM undergoes a type of thrashing wherein by the time the model has adjusted to the current state of a given behavior the behavior has already changed and the model is again incorrect. It is important to note, that if prior knowledge or a previous dataset is known, the model parameters such as refresh rate can be tuned to best fit the frequency of a given behavior. However, this would have to be done for every behavior, or at least a set or class of given behaviors and a method to accomplish this on a large scale or automatically does not exist. Finally, in regard to performance in respect to frequency DMM fails to accurately predict behaviors that have periods that do not fall on month boundaries. This is behavior is sim-

ilar to what happened in the door B experiment. Additionally, there is a visible difference between historical and future prediction accuracy for the trash sections, which drop from 91.13% to 32.13% in the trash 0 node. Once again, this is not a direct failure of the DMM model itself, but a result of forcing DMM to make long term predictions. Its historical predictions are much more accurate to its real-world behavior as DMM is more like an online learning model, changing it's predictions after every few observations, than an offline learning model that trains on a set of data.

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	92.31%	55.24%	96.17%	99.83%
Prediction Accuracy	68.92%	55.24%	95.97%	99.87%
Computation Time (Milliseconds)	610	60	90	6790
Memory Usage (KB)	31032	35660	35520	38372

Table 6.7: Hallway Delivery Section

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	91.13%	61.49%	64.52%	64.52%
Prediction Accuracy	32.26%	64.05%	63.21%	67.74%
Computation Time (Milliseconds)	600	60	70	1140
Memory Usage (KB)	31120	35364	35552	37192

Table 6.8: Hallway Trash Section 0

	DMM	Gaussian	FreMEn	Hypertime
Historical Accuracy	90.96%	61.09%	67.74%	67.74%
Prediction Accuracy	35.79%	61.09%	67.74%	67.74%
Computation Time (Milliseconds)	610	60	70	2120
Memory Usage (KB)	31116	35400	34928	37520

Table 6.9: Hallway Trash Section 1

6.2.2 Terminology and Metrics

In order to accurately and fairly compare the path planning results, terminology was devised to describe the mistakes made when planning with the use of a mod-

els predictions. Errors encountered when attempting to execute a path produced using a models predictions were grouped into two categories: hard errors & soft errors.

Hard errors are errors that cause issues that are impossible for a robot to theoretically recover from. This only happens when the ground truth path and the model predicted plan are at odds. That is to say, hard errors are when the ground truth claims a path is not currently possible and the predicted model claims there exists a valid path or vice versa.

Soft errors are errors that occur when the ground truth path and the model predicted path both agree on the existence of a path, but disagree on the specifics. This means that soft errors only occur when there exists a path to the goal, but the model predicted path is either too long or would cause the robot to run into an area with obstacles. However, by evaluating the ground truth it can be known that a path is possible and thus the robot in theory could create another plan with this newly discovered information. This newly planned path would then allow the robot to reach it's goal.

Assuming that the path produced by the model is valid, the path length is calculated and then compared against the ground truth path. Since both hard and soft errors produce invalid paths, they are not used to calculate the number of average cells traveled. This number is used as a quick and easy way of calculating the efficiency of paths generated when using a given spatio-temporal world model.

As mentioned in the previous section, there is no noise introduced into the simulated data. Thus, aside from DMM, the historical and future predictions are, for all intents and purposes identical. The reasons for this will be discussed in the section below, but because of this, only the future path planning results will be analyzed.

Finally, it is important to note that although all of the behaviors were combined to produce a path prediction the models were not simultaneously trained and asked to predict the various behaviors. Instead, each behavior, similar to the door experiment, was predicted individually and sequentially. Therefore, in order to have an overview

of the resources used, the total time spent on planning was calculated by summing up each of the individual times it took to train a specific model. Due to the fact that this was happening sequentially, not simultaneously, memory usage can then be taken as an average or a maximum over time. A maximum was chosen as it is assumed that memory is a finite resource that does not change over time and therefore as long as a system has the minimum amount of memory available to meet a programs' maximum demands there will not be any issues during run-time. In the future, if this was turned into a multi-threaded process the sum may wished to be used. A naive and safe assumption if this figure is desired otherwise is to simply multiply the maximum amount of memory used by the number of objects for which a predictive model is desired.

	DMM	Gaussian	FreMEn	Hypertime
Percent Hard Errors	60.15%	81.25%	50.00%	50.00%
Percent Soft Errors	14.92%	2.08%	2.69%	2.69%
Average Additional Cells Traversed	5.74	6.34	3.12	3.12
Total Time Spent Planning	3650	360	440	13780
Maximum Memory Usage	31384	35660	35588	38372

Table 6.10: Future Path Planning Results

6.2.3 Path Planning Results

Figure 6.10 shows the various models' performance metrics over simulated time. It is the goal of the models to reduce the number of errors and additional distance traveled when providing information for path planning. Thus the methods with the lowest lines on the graph are the most preferable methods. Using this general rule, it is clear that the Fourier methods have achieved the best results with a tie for both hard and soft errors amongst themselves. They are beaten out by Gaussian Mixture Models for soft errors, but this is only due to the significant number of hard errors that the Gaussian Mixture Model achieved. With so many hard errors, the method didn't have the opportunity to commit as many soft errors. Additionally, because hard errors imply the complete failure to predict if the goal is achievable they are

substantially worse.

One notable effect of using each of the multiple behavior predictions to do path planning is that there is an averaging effect over the data. Individual mistakes or incorrect predictions don't always make or break a plan. One place this is extremely obvious is in DMM's prediction of high frequency events. Despite the disparity between historical and future predictions for DMM in terms of these individual behavior prediction, the only difference in the final metrics are two additional soft errors during future predictions. This may be because of the large number of hard errors encountered preventing more soft errors from being generated, similar to the Gaussian Mixture Model. Another factor for this smoothing behavior is the location of the various behaviors and how they relate to a plan being possible.

In terms of computational cost, the path planning results appear to roughly scale linearly with the previous individual experiments results with minor deviations. The maximum memory used hovers between 30 to 40 megabytes for all of the experiments much like before. Similarly, DMM takes an order of magnitude longer than Gaussian Mixture Models and FreMEn to produce results. Hypertime again takes another order of magnitude longer than DMM taking almost 14 seconds However, a large disparity is visible when looking at the sum of time taken to produce a path. Despite taking less than 30 times as long to produce results, FreMEn achieved the same prediction and path planning performance as Hypertime.

6.3 Busy Elevators

6.3.1 Classification Methods

Upon even a cursory glance at the resulting graphs, such as the one in figure 6.12, it is immediately obvious that each model has a different approach to classification and prediction of the data. DMM, which predicts by averaging each of its sub-models, is perhaps the easiest to understand at first glance. It results in a somewhat smooth

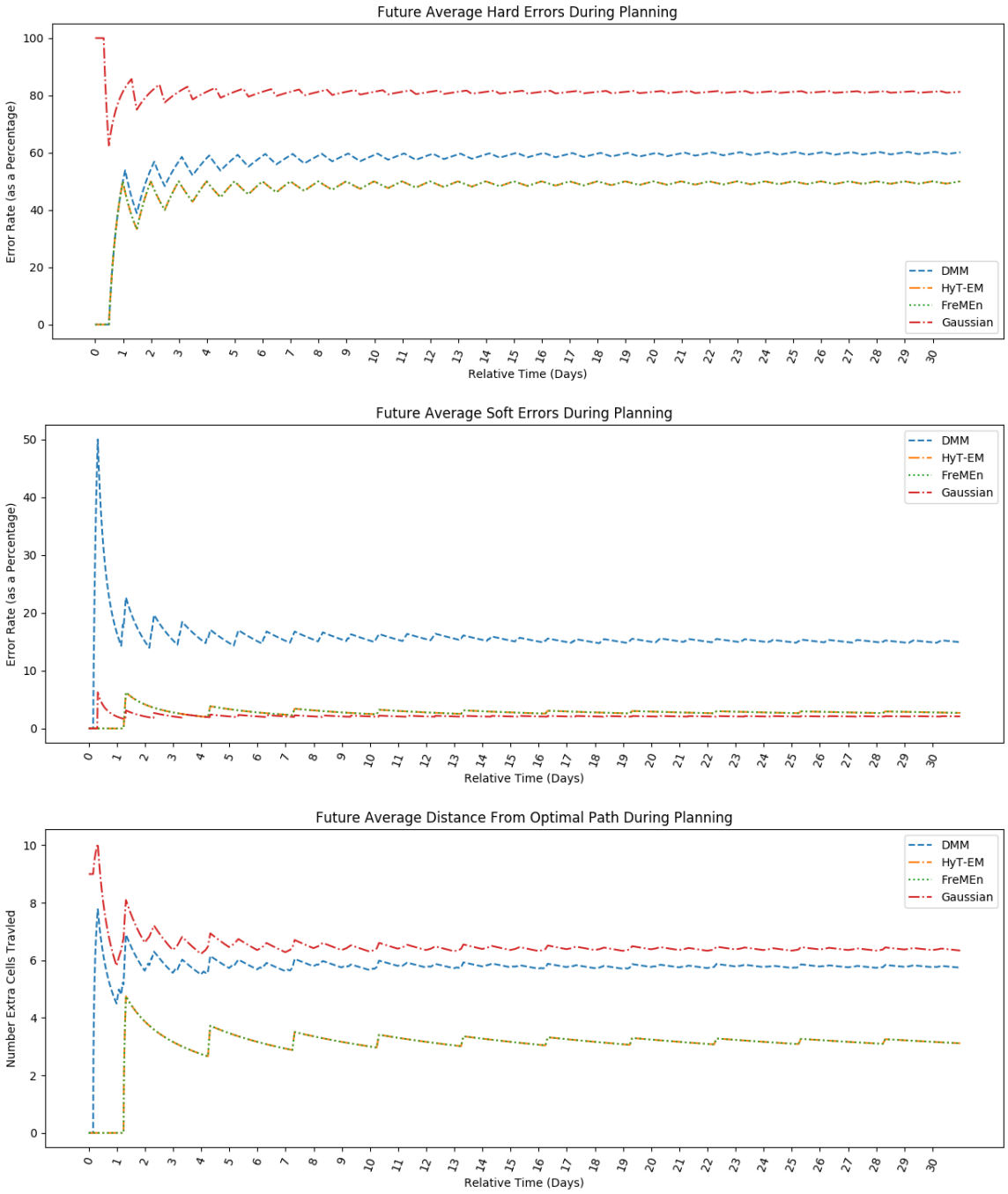

Figure 6.10: Future Planning Results

curve that follows the training data extremely closely.

On the other hand, Gaussian Mixture Models and FreMEn take a drastically different approach. Since, at their core they are binary classifiers, the prediction methods' as described in Section 5 result into two distinct prediction options. Both method's predictions hover around 0 and 7, although not exactly on an integer boundary. It is clear the lower value is close to 0 due to the large amount of time the elevator spends at what is essentially a rest state with next to no use. This is mostly between the hours of 22:00 and 08:30 or 10.5 hours. The other prediction that hovers around 7 is slightly harder to predict with precision but can be roughly estimated when looking at the nominal times for waiting. When ruling out values that would be placed into a 0 prediction, there are a large amount of nominal values between 4 and 9 which average out to 6.5. Finally, due to the large amount of noise combined with the fact that an elevator cannot have a negative wait time, the average is pulled up a bit by the extremely long wait times that can be as high as 30 seconds during peak hours.

Finally, and perhaps most interesting, are the Hypertime results. Hypertime takes a unique approach to classifying its data. Once again, looking at the Hypertime graph in figures 6.11 and 6.12 it is seen that predictions are clustered on integer boundaries. To be clear, non-binary Hypertime in its current implementation, although capable of training on real data[1], can only produce integer predictions. This limitation is due to how predictions are evaluated before being accepted as the actual prediction for a given time. When producing a prediction, or an estimate as referred to in Hypertime, "the likelihood distribution of the sample having a given value at time t" [2] is calculated for each sample from 0 until a maximum value. A maximum value for 30 was used for this experiment. This means that during calculation Hypertime would never be able to predict a wait time of 30 seconds or greater. Looking again

[1]Real data as in both real-world data and all the numbers in the set of real numbers that can be represented using standard floating point notation

[2]This is a comment taken from the Hypertime source code. Particularly the estimate function in CHyperTime.cpp

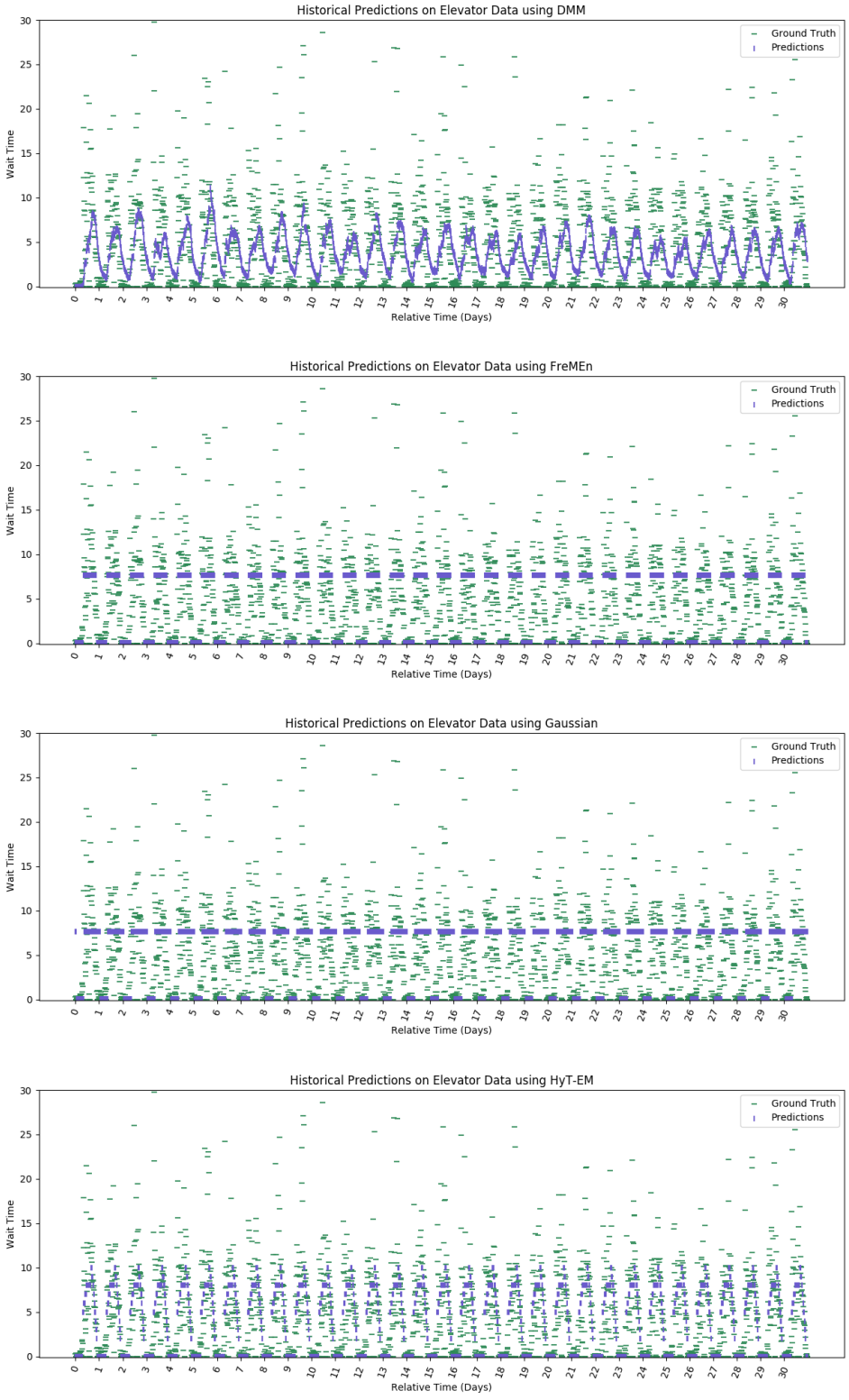

Figure 6.11: Historical Recreations - Elevator Data

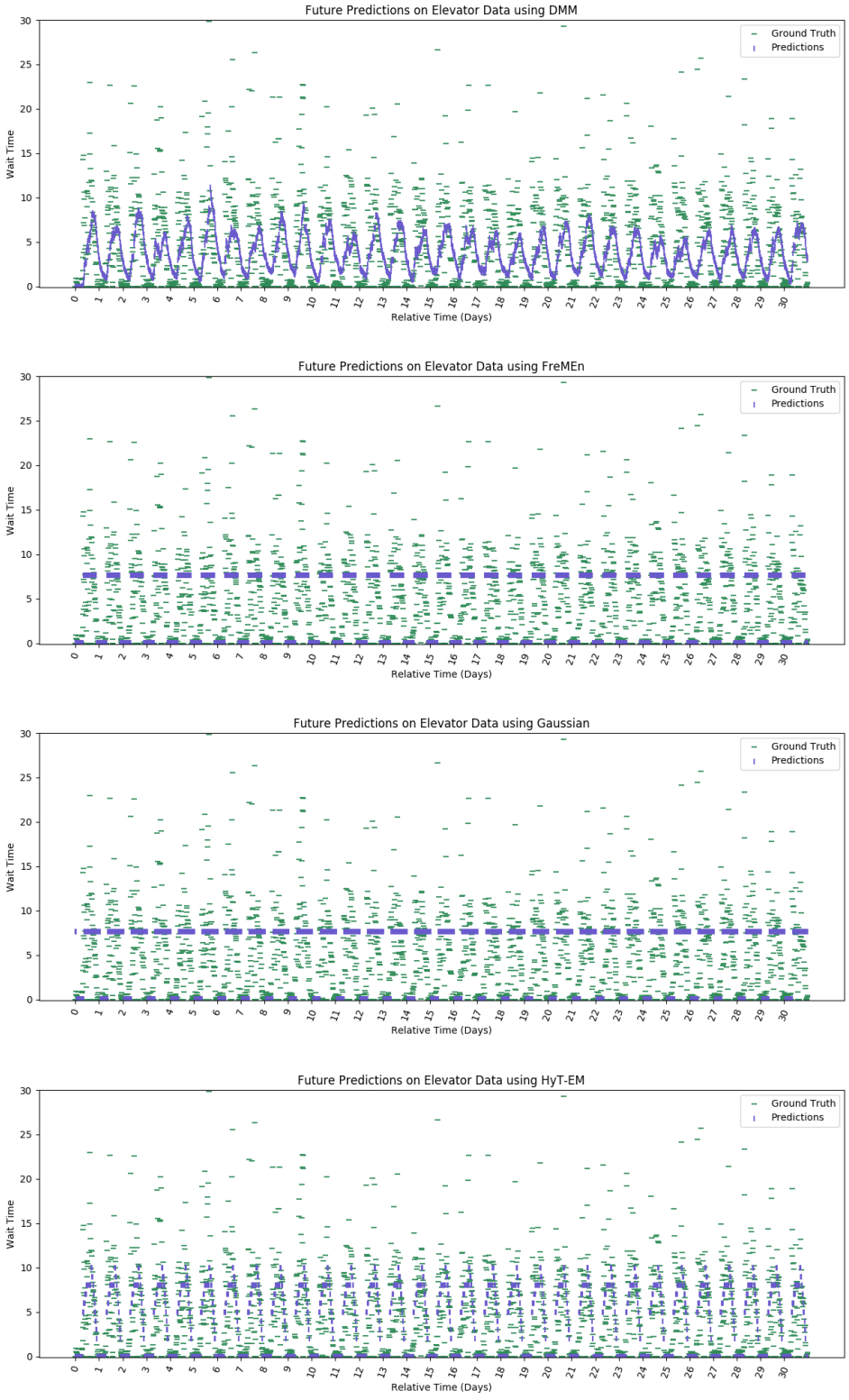

Figure 6.12: Future Predictions - Elevator Data

at the resulting figures, it is clear this is not an issue as there does not appear to be any prediction greater than 10 seconds. Thus, with no predictions even nearing 30, it is reasonable to conclude that the upper limit did not hamper the ability for the model to produce accurate predictions.

6.3.2 Metrics

The only additional metric included in this experiment was that of "Average Distance from Correct Wait Time". This, as the name implies, simply takes the difference between a models predicted wait time and the ground truth wait time provided in the data. The absolute value of this difference is saved and then averaged for every time step to illustrate how the average changes over time. This metric does not make a differentiation between the robot waiting for the elevator versus the elevator waiting for the robot. No other additional factors have been taken into account. Specifically, this includes something like calling an elevator so far in advance that the elevator shows up and is called by someone else.

6.3.3 Performance & Scalability

	DMM	Gaussian	FreMEn	Hypertime
Historical Average Additional Wait Time (Seconds)	3.05	3.24	2.13	1.90
Future Average Additional Wait Time (Seconds)	3.06	3.25	2.12	1.92
Computation Time (Milliseconds)	610	60	70	7690
Memory Usage (KB)	31092	34636	34672	67788

Table 6.11: Elevator Wait Time Overview

Table 6.11 shows similar results to those in previous sections with a few notable exceptions. Foremost, computation time remains mostly consistent, with Gaussian Mixture Models and FreMEn coming in at the upper tens of milliseconds, DMM an order of magnitude higher and Hypertime being yet another order of magnitude higher than DMM. Hypertime, although remaining under 10 seconds, does take slightly longer than would be expected compared to previous results. This

	DMM	Gaussian	FreMEn	Hypertime
Historical Average Additional Wait Time (Seconds)	1.58	4.47	2.05	1.58
Future Average Additional Wait Time (Seconds)	1.65	4.47	2.05	1.59
Computation Time (Milliseconds)	7350	90	350	250037
Memory Usage (KB)	75304	35140	34980	79864

Table 6.12: High Resolution Elevator Wait Time Overview

extra time is likely to originate from the switch from binary to non-binary state classification. As mentioned previously, when doing non-binary classification Hypertime must check every integer value from 0 until a maximum prediction value, which in this experiment was 30. Had this number been reduced to something closer to 10 (the maximum experimentally predicted value) it is likely that the computation time would have been reduced to 2 or 3 seconds placing it closer to previous experiments but with a slightly longer time due to the additional calculations needed. This same trend and effect is mirrored in memory usage for the same reasons.

Finally, prediction wise, once again similar results are visible. Hypertime has the best results with under 2 seconds of average distance from the correct wait time, FreMEn trailing slightly behind at barely over 2 seconds, DMM and then Gaussian both over 3 seconds.

Overall, the results are not particularly surprising when looking at previous experiments, until the results in Table 6.11 are compared with the results in Table 6.12. The High Resolution data, sampled at a rate of once per minute instead of once per 15 minutes, highlights and exemplifies the data found in the lower resolution data.

Perhaps the most obvious and most interesting change is the drastic improvement in DMM, which halved its wait time error over the lower resolution data. The main conclusion to be drawn from this is the importance of tuning DMM to refresh its internal representation of objects with the expected occurrence of periodic changes with respect to the rate at which that data is being sampled. DMM struggles to predict changes in daily behavior when samples occur at 15 minute increments as seen both in the low resolution data for the elevator and the meal nodes back in the navigation experiment. Additionally, a tenth of a second decrease in wait time

prediction is now much more obvious. In the real-world this would not be a significant problem, but it shows issues with long-term future predictions.

Figure 6.13 shows an overview of the different models over time. Slight movement was seen in either direction for Gaussian Mixture Models and FreMEn. The simplistic nature of the Gaussian Mixture Models results in roughly a 1 second decrease in prediction accuracy, but it does manage to maintain resource usage very close to that of the lower density data. This suggests that Gaussian Mixture Models may be a good fit for very large datasets with simplistic periodic behavior where precise accuracy is not necessary. FreMEn on the other hand had a slight increase in prediction accuracy but saw a slight increase in computation time while maintaining memory usage. This increase in computational time, specifically when compared to Gaussian Mixture Models, likely comes from the model order FreMEn operates at (3), which causes three loops through the training data for predictions. Since the loops are merely over the same training data, but with different parameters for estimation it would make sense the computation time would increase while memory usage would stay relatively constant. In fact, a rough calculation shows just this. Gaussian Mixture Models take about 30 milliseconds longer in the high resolution data, an increase from 60 to 90. Adding an additional 30 milliseconds to FreMEn's low resolution computation time value of 70 yields 100 milliseconds. Accounting for the additional work done by FreMEn by multiplying by 3 yields around 300 milliseconds. This is close to the observed 350 milliseconds, especially when additional overhead is considered.

Finally, Hypertime has arguably the most intriguing change between high and low resolution elevator data. There was only a relatively small increase in memory usage. The computational time, however, saw an extremely large increase to over 250000 milliseconds, or over 4 minutes. This additional run-time only yields a performance increase of between 300 and 400 milliseconds. It may be argued that the relatively small increase of prediction accuracy may be the result of Hypertime being unable to make more accurate predictions due to its integer limitation. Without this limitation it could perhaps continue to increase in accuracy but this would only further increase

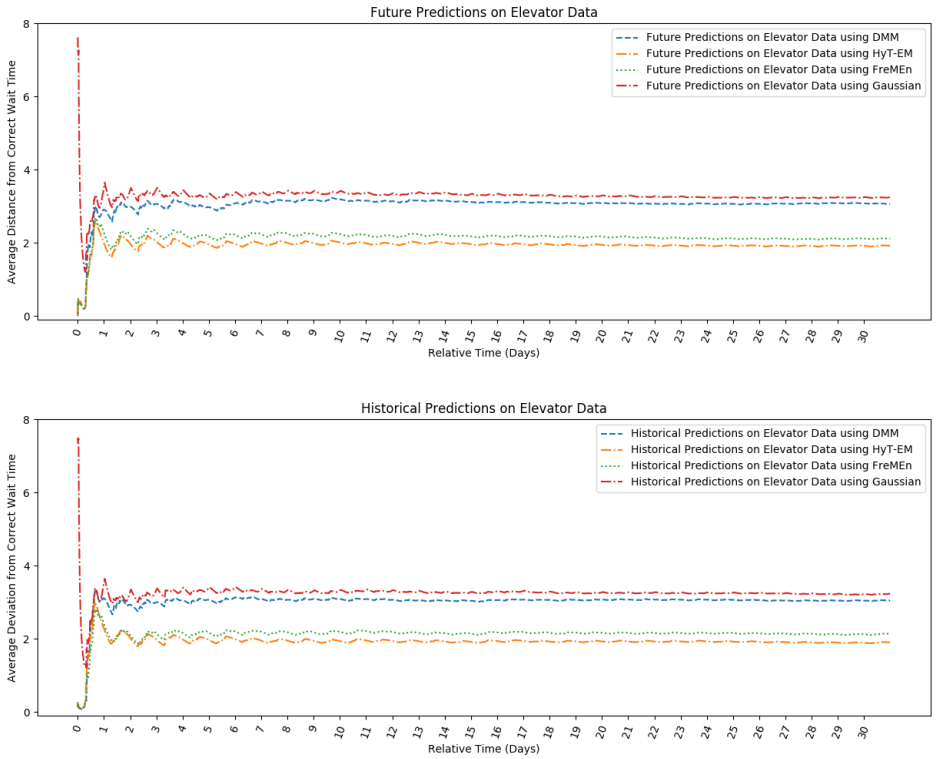

Figure 6.13: Model Accuracy Over Time - Elevator Data

its computational time. The computational time is likely to have scaled they way it did because of the large number of calculations done for each data point during the estimation portion of prediction. As noted previously, in this specific case 30 complex calculations must be done before a single estimation can be chosen at a given time t. It is important to recall that a reduction in the maximum prediction value, would most likely reduce the computational time considerably, perhaps as much as a third of its original value if the max estimate was limited to 10. This would likely result in a computational time between 1 and 2.5 minutes.

6.4 Final Thoughts

After reviewing the 3 experiments and 4 methods several trends can be identified. Starting with computational efficiency, memory usage appeared relatively consistent through all experiments and methods. With just under 3000 data points per test month and around 30KB of memory used per run it appeared that each data point takes slightly more than 10 bytes plus some overhead. This represents the minimum cost for something like DMM although other methods use slightly more memory. Specifically, when ranked from highest to lowest memory usage, the order was Hypertime, FreMEn, Gaussian, and DMM.

With respect to computational usage in the time domain, a different pattern is observed. Ignoring minor fluctuation between experiments, Gaussian and FreMEn consistently used tens or hundreds of milliseconds on 3000 data points. DMM used an order of magnitude beyond that, and Hypertime used yet another order of magnitude more. Although this is the average case it would appear that scaling is not linear for Hypertime as demonstrated in the high resolution elevator experiment. Future testing will have to be done to determine the scaling factor.

In terms of accuracy, the various methods each had their own strengths and weaknesses. The Gaussian method was on average a very efficient technique and consistently was the least accurate when making predictions. It appears the Gaussian method was only able to make semi-accurate predictions when presented with data that was highly regular, contained minimal noise, and had a relatively high frequency

of occurrence with respect to the observed time frame. Osculations are also visible in a large number of the accuracy graphs, especially in the earlier sections of the graph. These osculations do not originate from any of the models behavior but instead are a result of how the averages were computed. Averages were computed over time such that, at the beginning, only a few data points existed, and thus a few right or wrong predictions have a large impact. As more an more data points are added to the average computation the graph will be less affected by correct, or incorrect, predictions and will tend to even out.

The Fourier based methods, FreMEn and Hypertime, fared much better with noisier and irregular data. This trend was particularly true when the data had a relatively high frequency was found in Laundry or Meal nodes in the path finding experiment. Despite having to overcome binary restrictions, FreMEn still fared relatively well on the elevator experiment and demonstrated a strong versatility. Finally, and perhaps most interestingly, DMM displayed a multitude of unique results. Performance was highly dependent on the frequency of the data involved. When the refresh rate of the multi maps was tuned closed to that of the frequency of object or behavioral changes, it preformed in a similar manner to that of Hypertime while using vastly fewer resources. However, when tuned to a non-optimal frequency, DMM can occasionally preform worse than even the Gaussian method.

7

Conclusions

7.1 Contributions

First and foremost, this work developed a comprehensive method for the comparison of various spatio-temporal world modeling methods. Although as some previous work in this area already exists, that work was limited both in its ability to be applied to various methods and the depth at which any two methods could be compared. The new techniques for comparison established in the Criteria for Comparison section built on the current methods to offer increased breadth and depth. Furthermore, the ROPOD case study section shows in detail how these new testing criteria can be used with respect to an actual project. This is an extremely important contribution as nearly all previous work merely used criteria to look reflexively at the quality of a new approach and not which method may be best suited for a given problem.

Building on the qualitative analysis work already presented in this paper, the experimental section provided guidelines for how one would select a set of experiments to test and refine previously obtained results. These experiments highlight unique aspects of the techniques presented that were not necessarily highlighted in their original papers. Additionally, this section provides a roadmap for what types of experiments should be done, what results to look for, and how to evaluate those results. Beyond a roadmap, a novel approach for applying binary classification to non-binary data was presented and demonstrated. Lastly, some new metrics were

proposed to evaluate path planning.

Finally, a common thread for both the qualitative and experimental sections was the emphasis placed on experiments being done with respect to a real-world project. Although, in the past some work has been done with real world projects, comparative techniques primarily focused on the performance of the existing specific algorithm under test. That is to say, previous work has focused solely on how "correct" a spatio-temporal modeling technique was for an individual object. This work focuses on additional information like computational intensity, and performance when working with the resulting predictions.

7.1.1 Recommendations for ROPOD

After paring down the initial list of methods for spatio-temporal world modeling, designing and running the experiments, and analyzing the results, it is time to make the final recommendations. An important note is that despite being included in the experiments, the Gaussian model will not be considered in the recommendations. This is because its inclusion was for comparison sake and not as a serious contender. Additionally, it also had consistently the worst accuracy. With that aside, the methods under consideration are dynamic multi-maps, FreMEn, and Hypertime. Dynamic multi-maps are a tempting option given their low computational usage and occasional moments of high performance and accuracy. Despite these upsides, the dynamic multi-maps are not recommended due to the high level of maintenance and tuning required to achieve high accuracy predictions. That leaves the final two methods, FreMEn and Hypertime. Both of these methods have fairly similar performance when making binary predictions with Hypertime having a bit of an edge in accuracy, but FreMEn edging Hypertime out in computational resource usage. However, this is not the case when working with non-binary data. In this case, Hypertime had a noticeable lead in prediction accuracy. This does not come with an additional penalty with respect to computational usage though.

In order to best weigh this trade off it is necessary to review the ways that ROPOD

will use spatio-temporal world modeling. Predictions will be used for determining optimal path planning by modifying a preexisting world model. Since both FreMEn and Hypertime use offline learning, the models can be trained overnight during a period of low usage. A central server will be used for data storage and training. This is advantageous because it allows computational power and resources be concentrated instead of each robot being individually responsible for processing. Centralization allows an older model to be stored on the central server and used while a newer one is trained. Because an old model can be used while a new one is trained, training time is less of a concern. Given it's high accuracy, and the central servers ability to mitigate some of the heavier resource usage, it is recommended that Hypertime be chosen for use with ROPOD.

Integration can be achieved by establishing a database on the central server where observations made by robots in the field can be stored along with a timestamp. At times of low ROPOD usage, most likely overnight, new models can be trained for usage during the next day. During this time, observations will be retrieved from the database, sorted, and used to make models for each of the desired objects or behaviors. That is to say, for doors, graph edges, etc. Once the new models are complete, whenever a query is made for a given time they will be asked to produce a prediction. Those predictions will then be used to modify the existing world model at which point path planning can take place. If, as the project scales, it is determined that Hypertime is too much of a computational drain FreMEn can be introduced for use with binary data to lighten the load.

7.2 Future work

Although this work has made helpful contributions to the field, it is certain that further work remains. ROPOD can be used as an example to base future studies. This is particularly true because ROPOD, unlike previous work, is a multi-robot system. This means that while running tests with respect to a single robot is valid and fully accurate, the addition of multiple robots creates additional opportunities and complexities. Having multiple robots moving about the environment at different times allows for gathering much more data. This in turn decreases the sparsity

of data available for testing and developing models. Furthermore, this is likely to increase model accuracy as evident in the elevator experiments, but may also cause issues when trying to mitigate or merge multiple samples of the same object at the same time. Moreover, the very presence of multiple robots may themselves introduce spatio-temporal complexities. Having multiple robots moving about a shared space, especially during peak traffic, could create additional obstacles in the form of congestion that would need to be planned around and accounted for in advance.

Another area of future work is the collection of additional real-world data. This data is often more noisy, irregular, and generally unpredictable when compared to even the best simulated data. Ideally this new data new data will be collected and compiled into a new dataset for use in other experiments. Early steps have been been taken as exemplified in the "Brayford" dataset [16]. This dataset, however, contains only 3D data for a single room. The observation was done using a static device and thus observations are at regular intervals and contain minimal noise. Furthermore, because this is was only one device the complexities of a multi-robot system are also overlooked. The work done in this paper is an excellent step in the right direction, but the field of spatio-temporal world modeling remains open with many rich and intricate problems left for exploration.

A

Congested Hallway Experiment Results

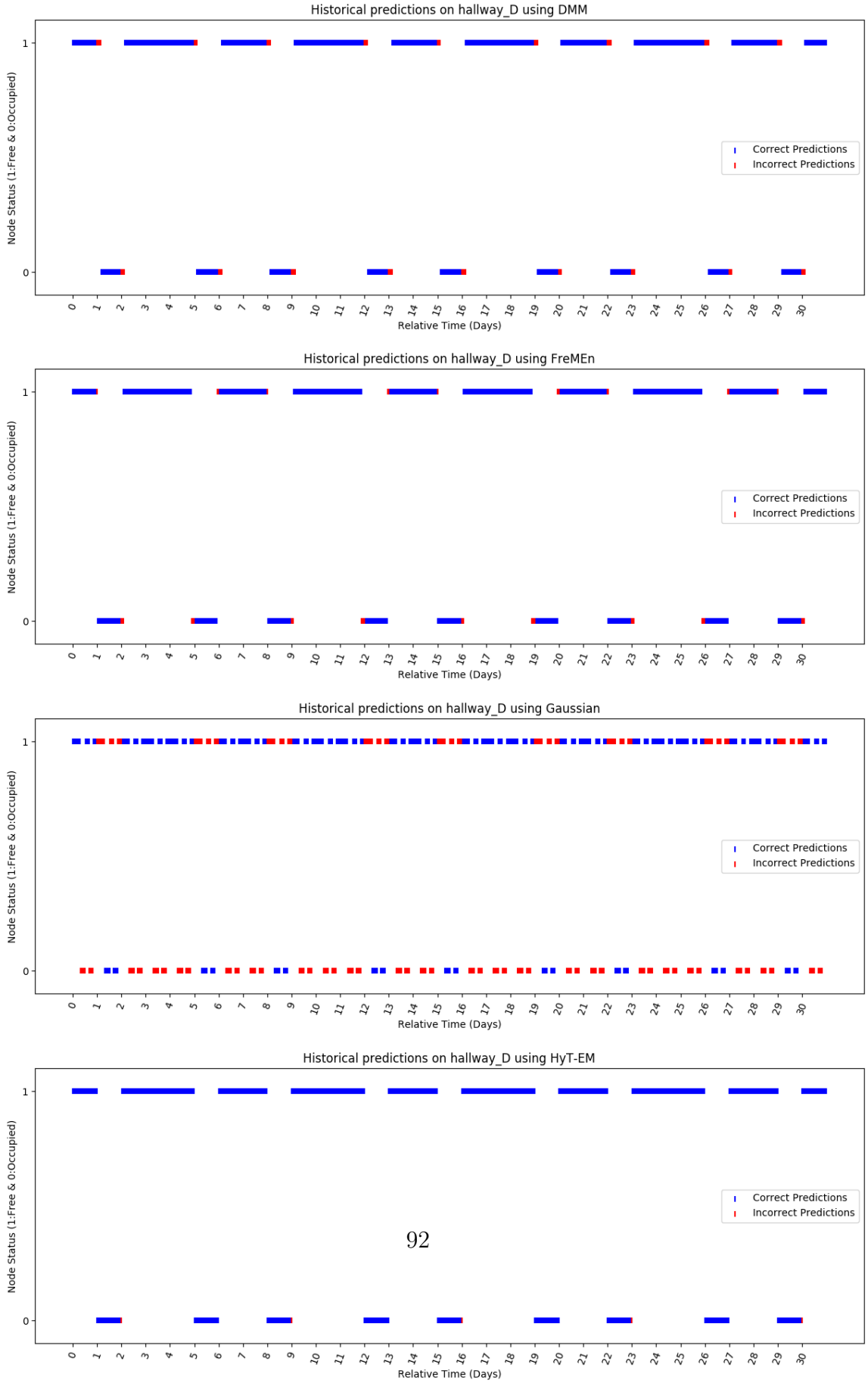

Figure A.1: Historical Recreations - Hallway Delivery

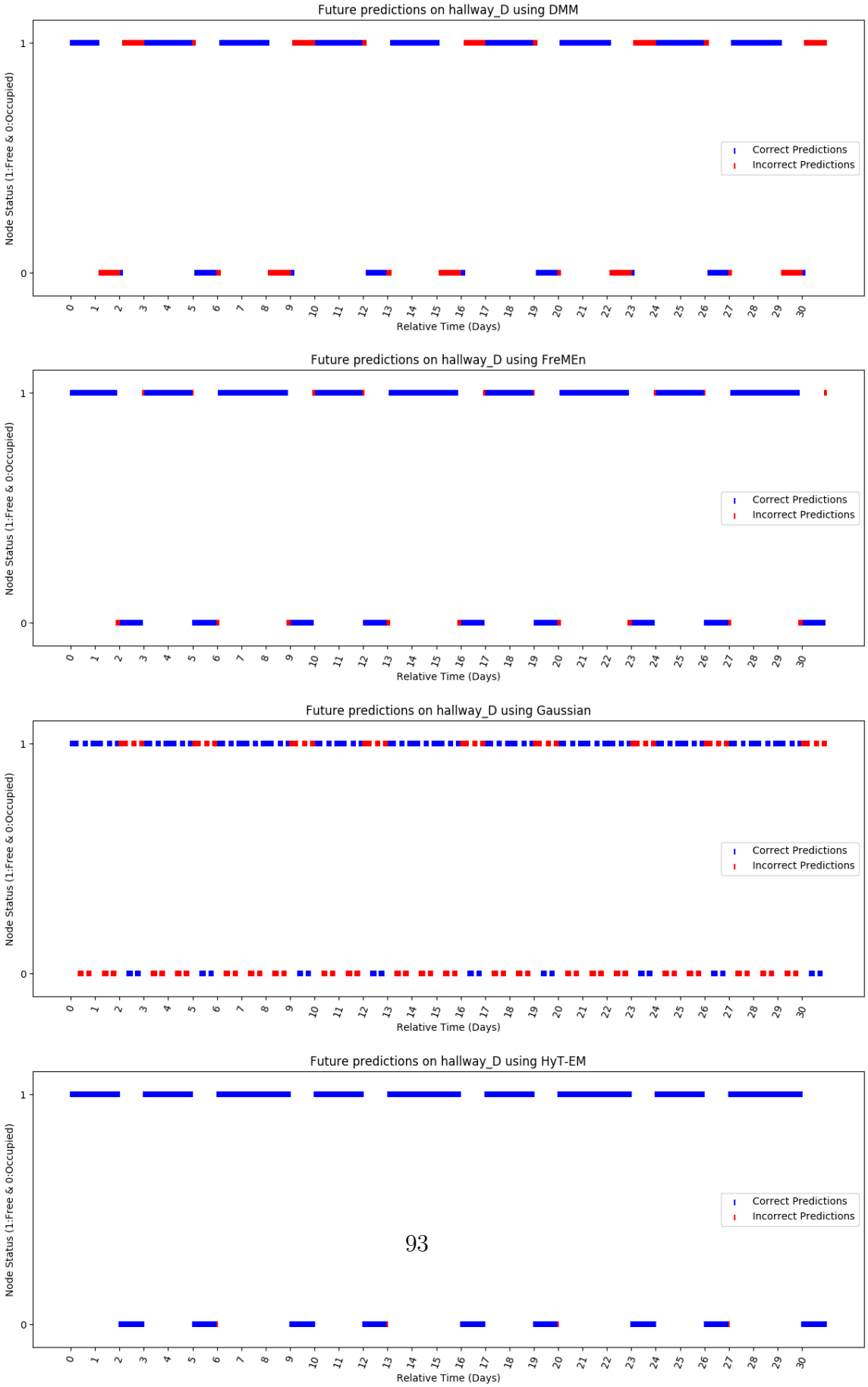

Figure A.2: Future Predictions - Hallway Delivery

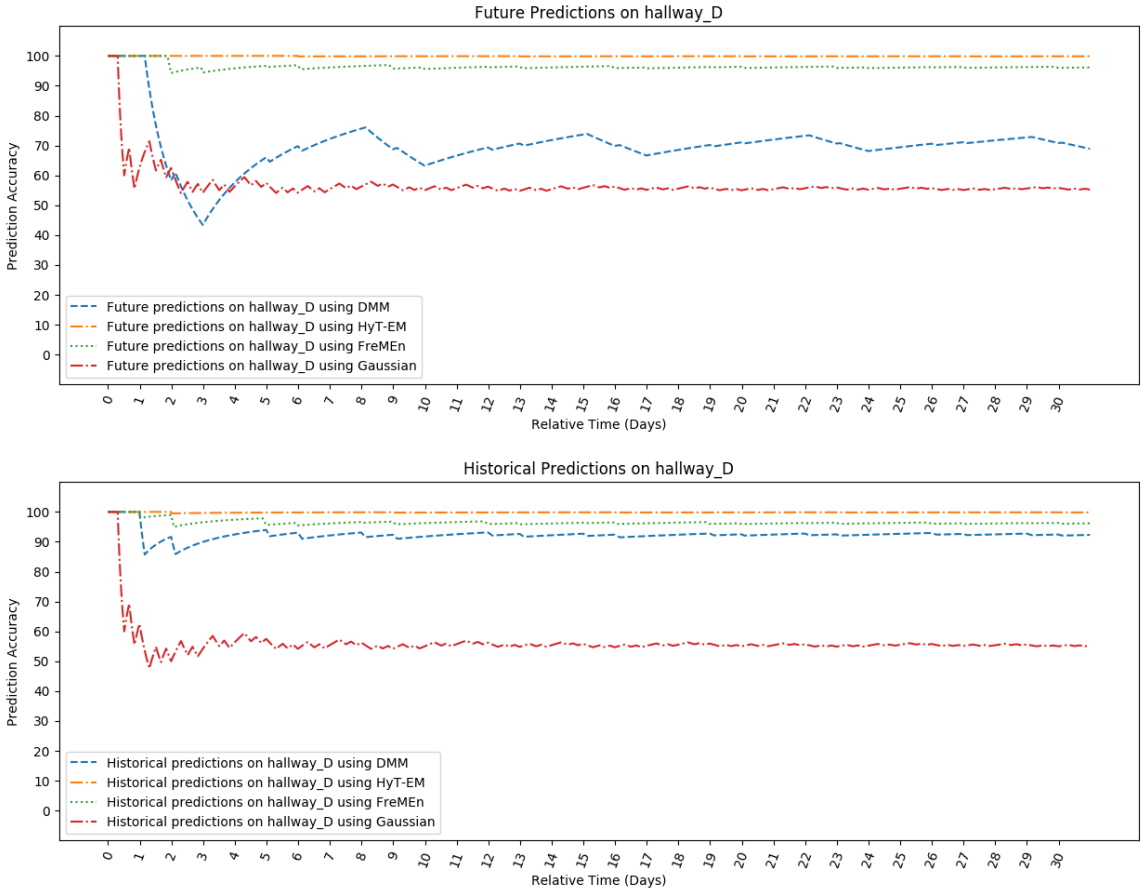

Figure A.3: Model Accuracy Over Time - Hallway Delivery

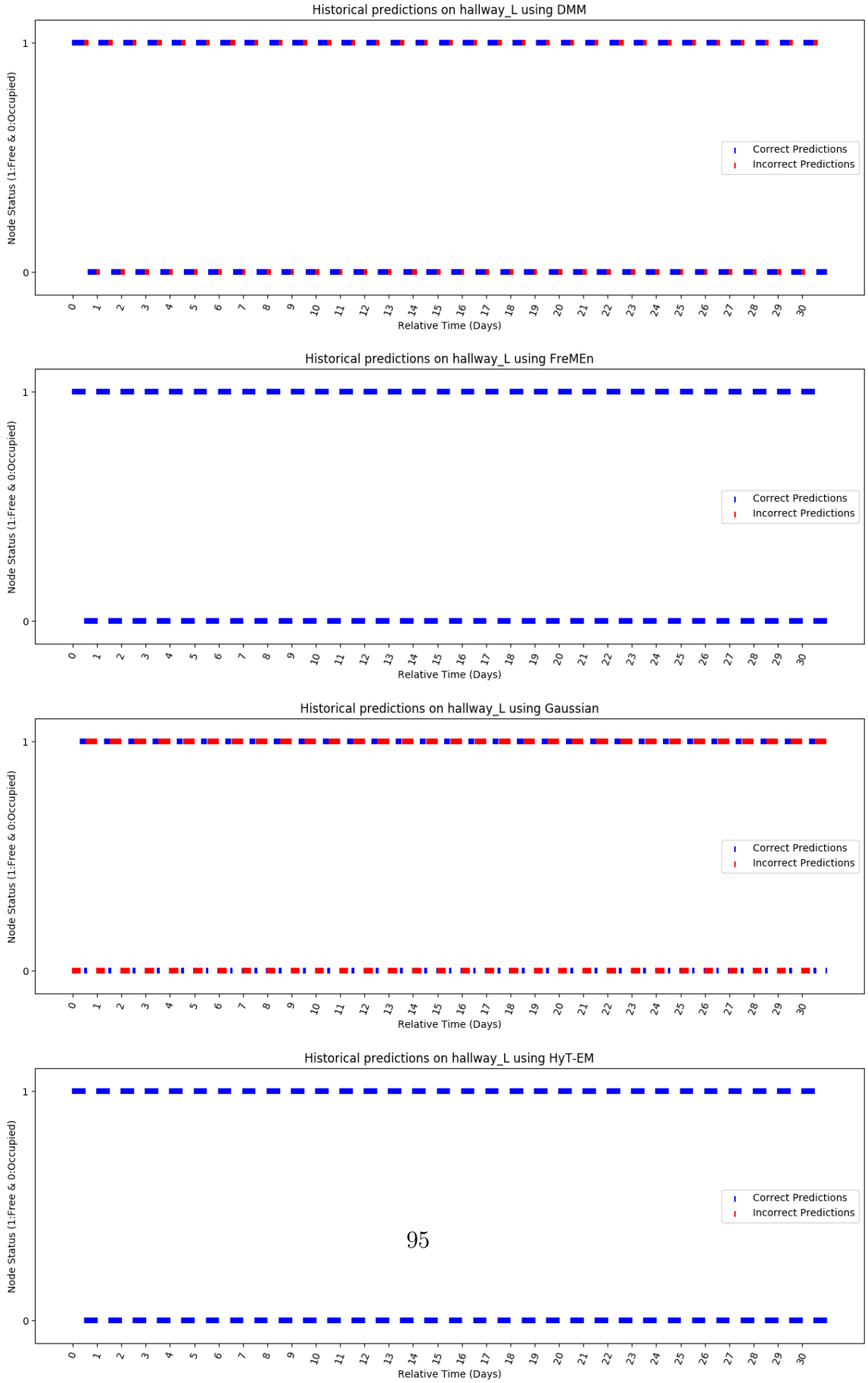

Figure A.4: Historical Recreations - Hallway Delivery

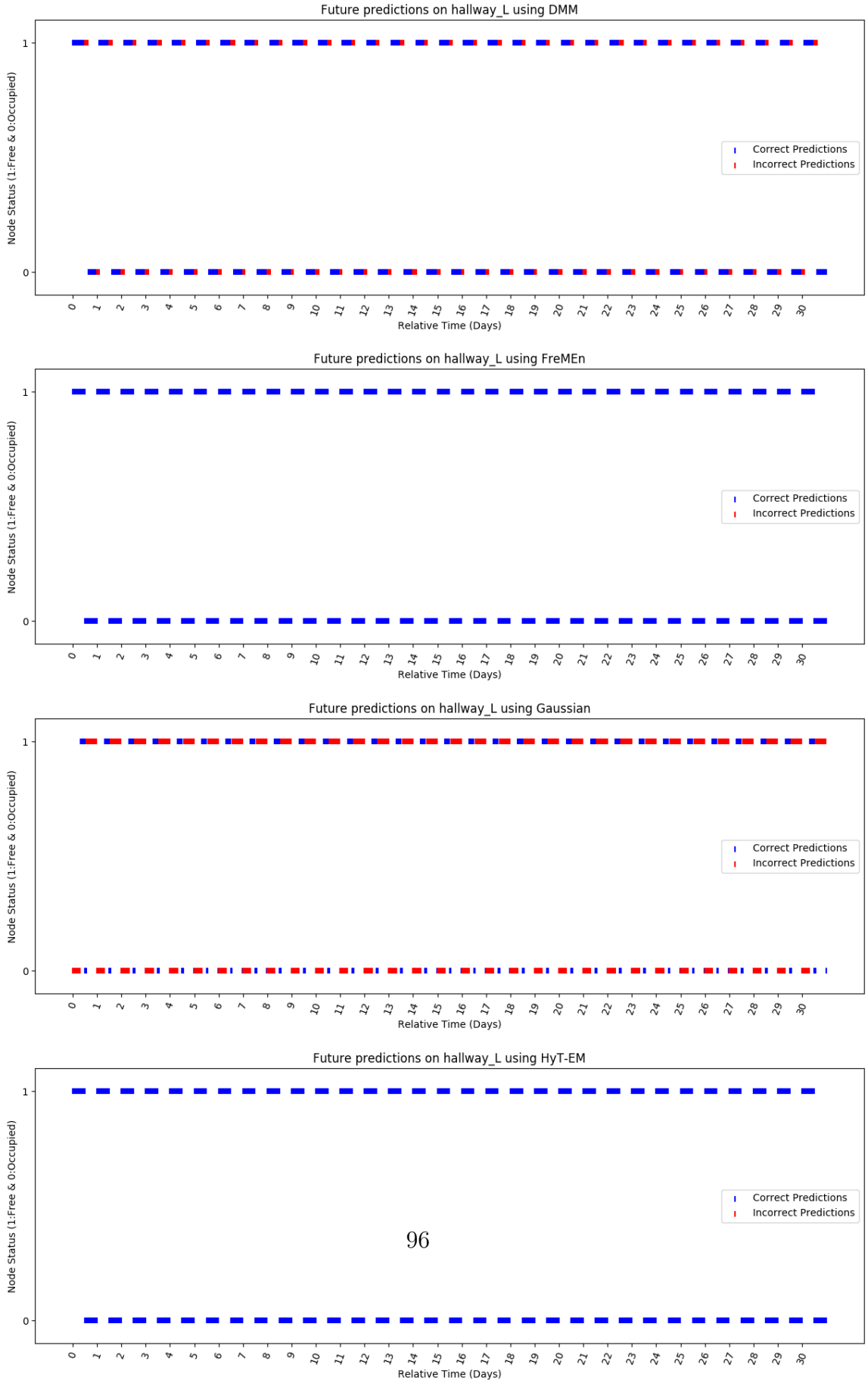

Figure A.5: Future Predictions - Hallway Laundry

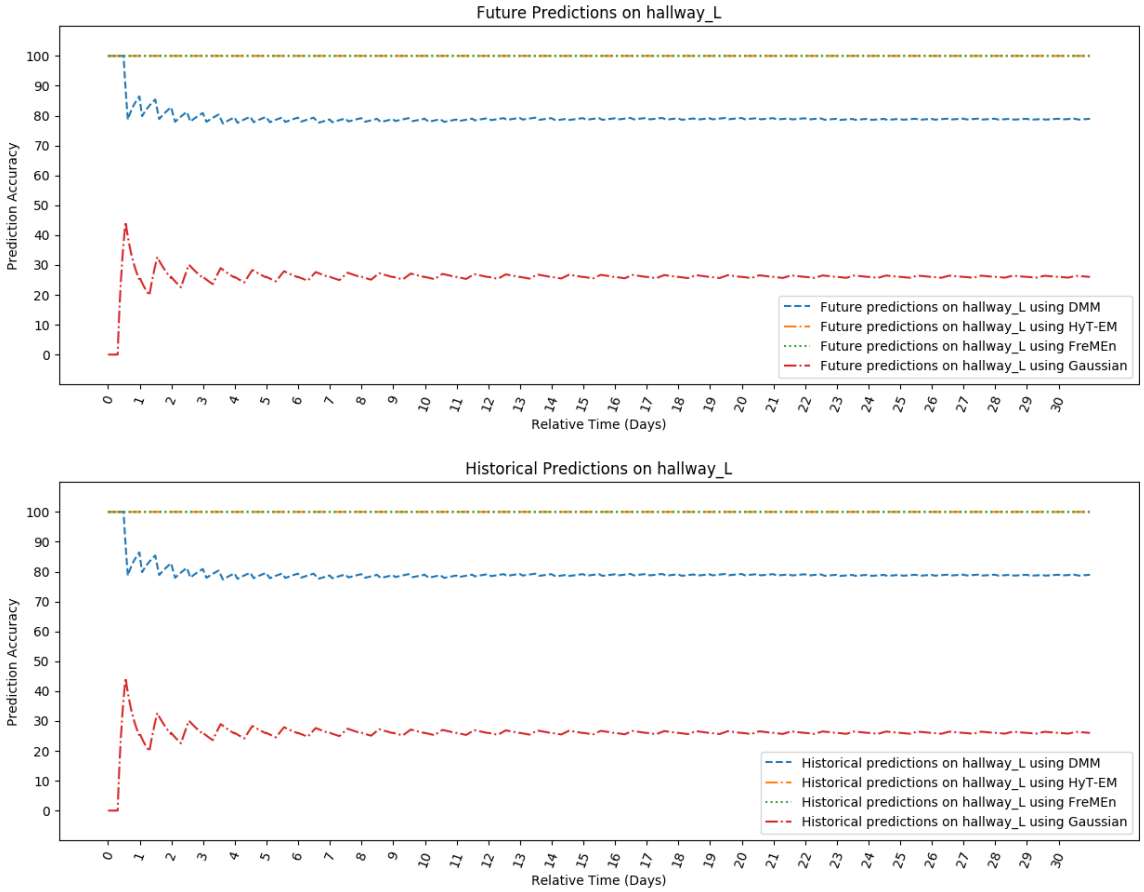

Figure A.6: Model Accuracy Over Time - Hallway Laundry

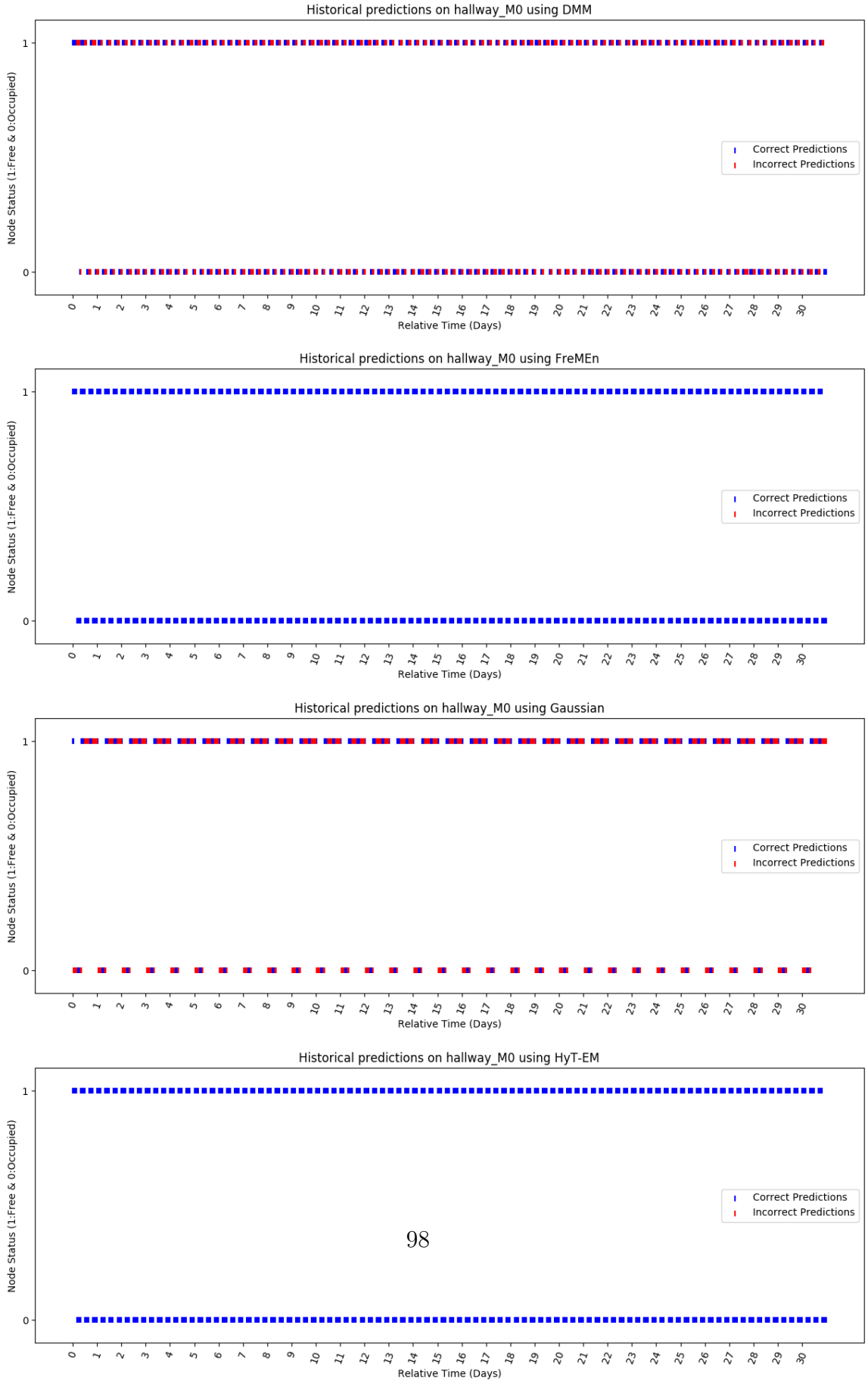

Figure A.7: Historical Recreations - Hallway Meal Section 0

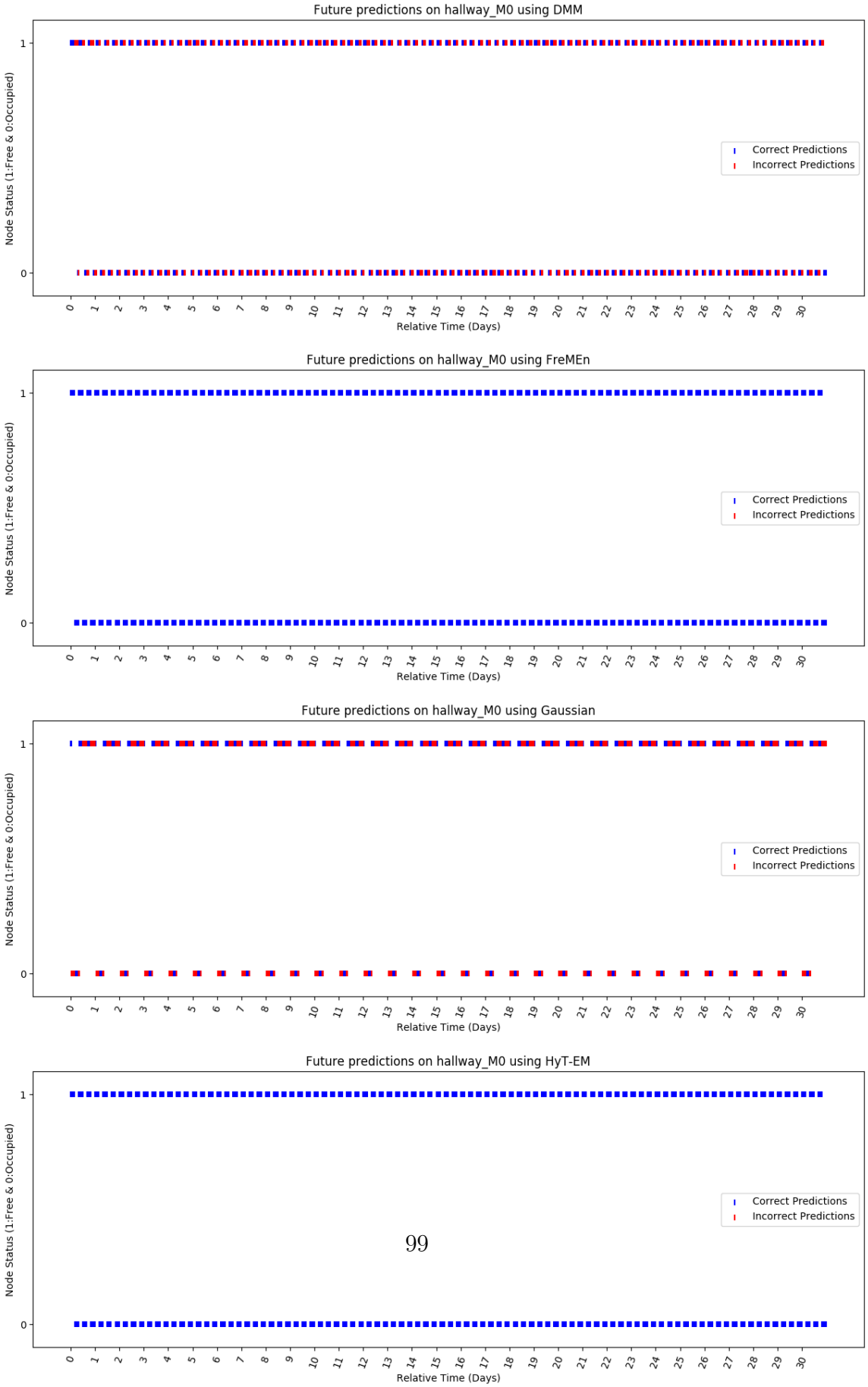

Figure A.8: Future Predictions - Hallway Meal Section 0

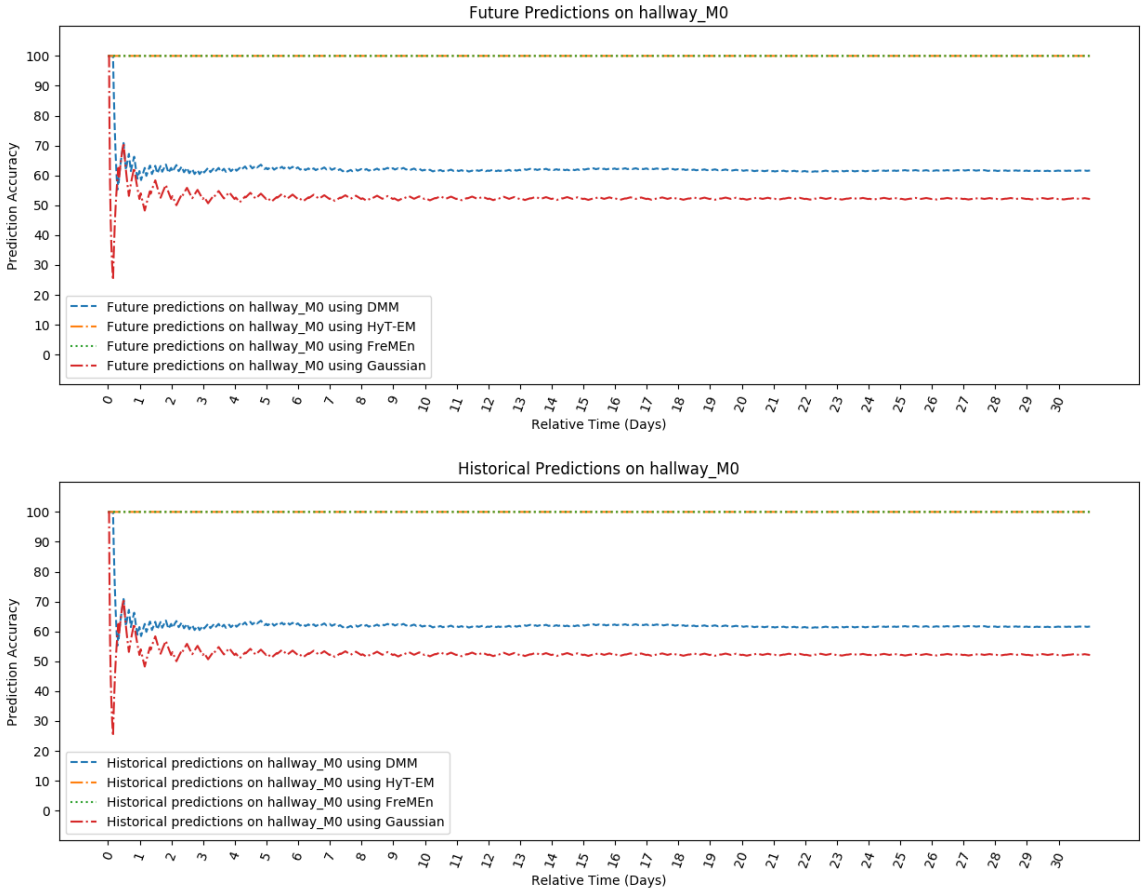

Figure A.9: Model Accuracy Over Time - Hallway Meal Section 0

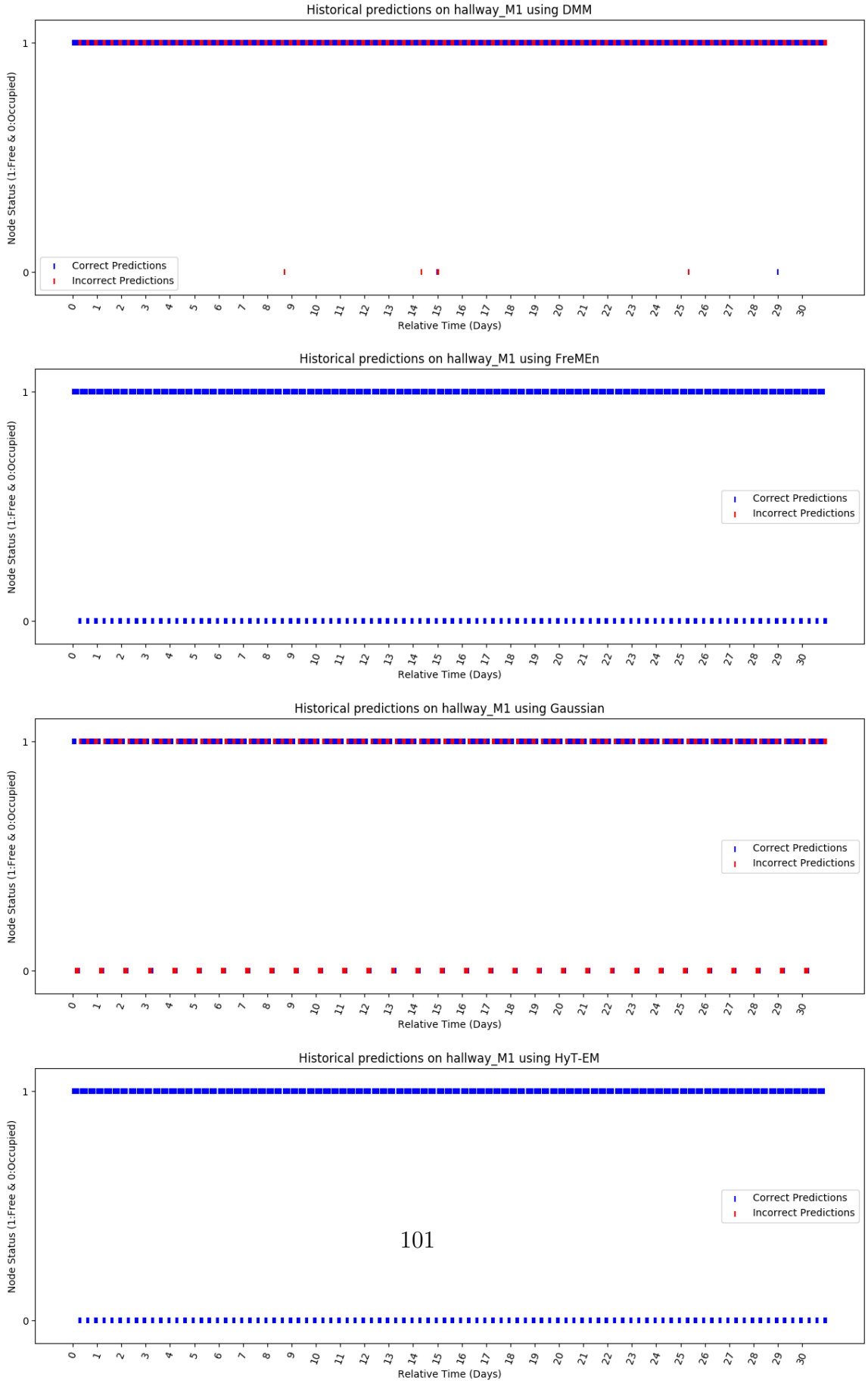

Figure A.10: Historical Recreations - Hallway Meal Section 1

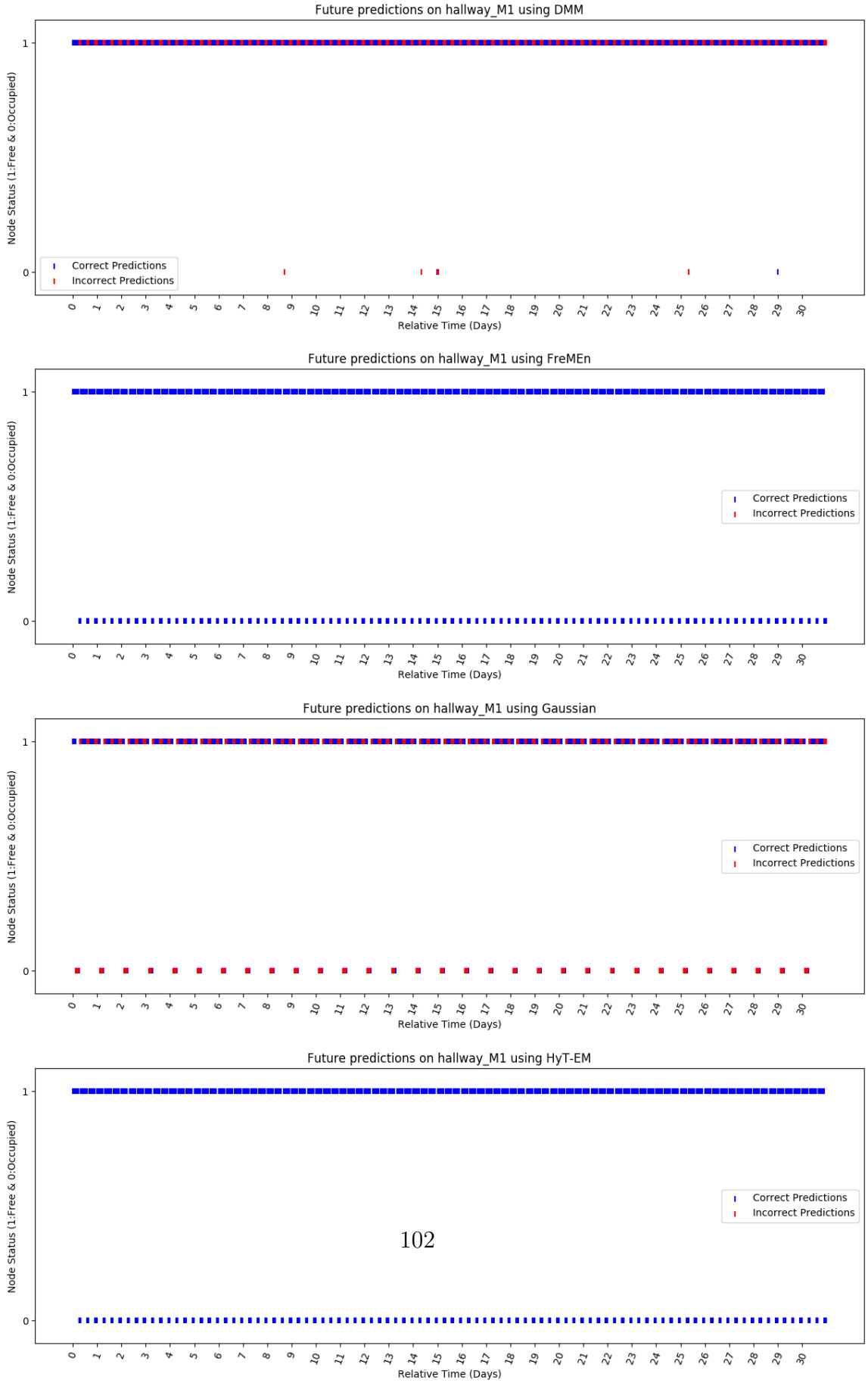

Figure A.11: Future Predictions - Hallway Meal Section 1

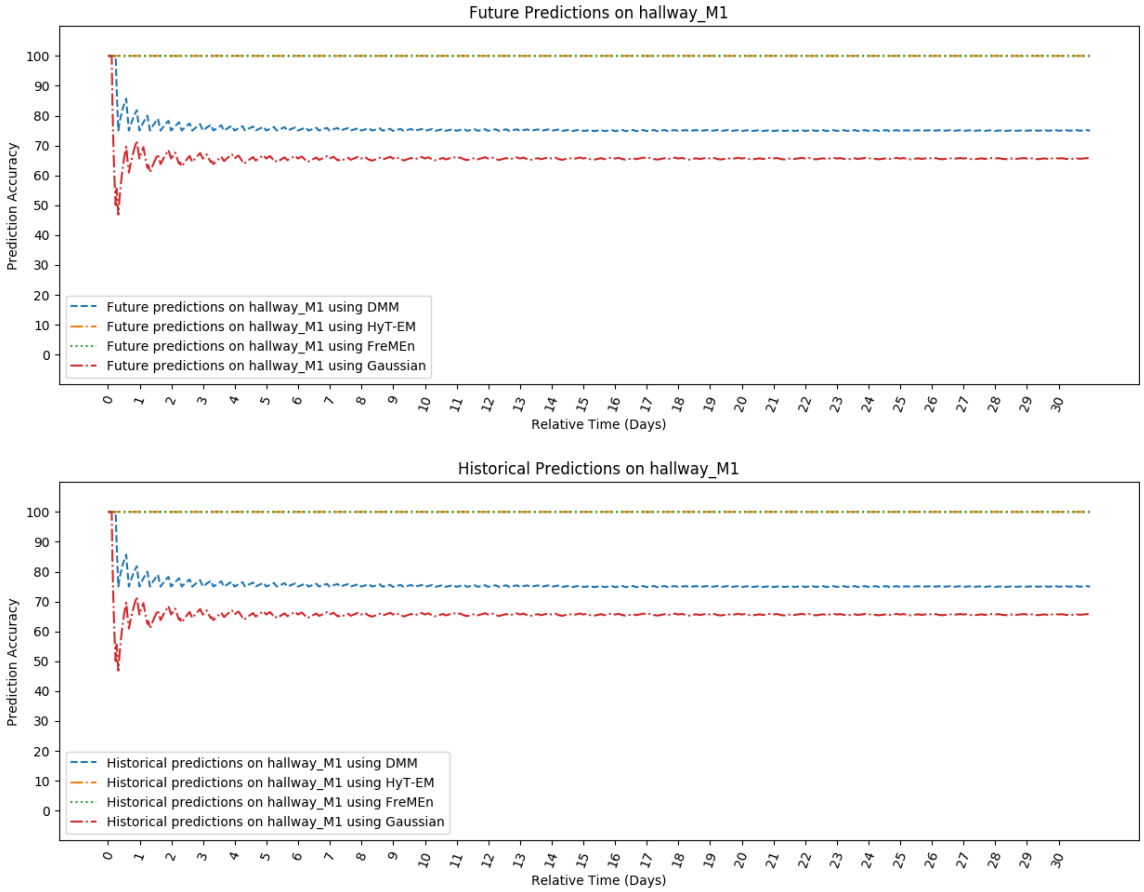

Figure A.12: Model Accuracy Over Time - Hallway Meal Section 1

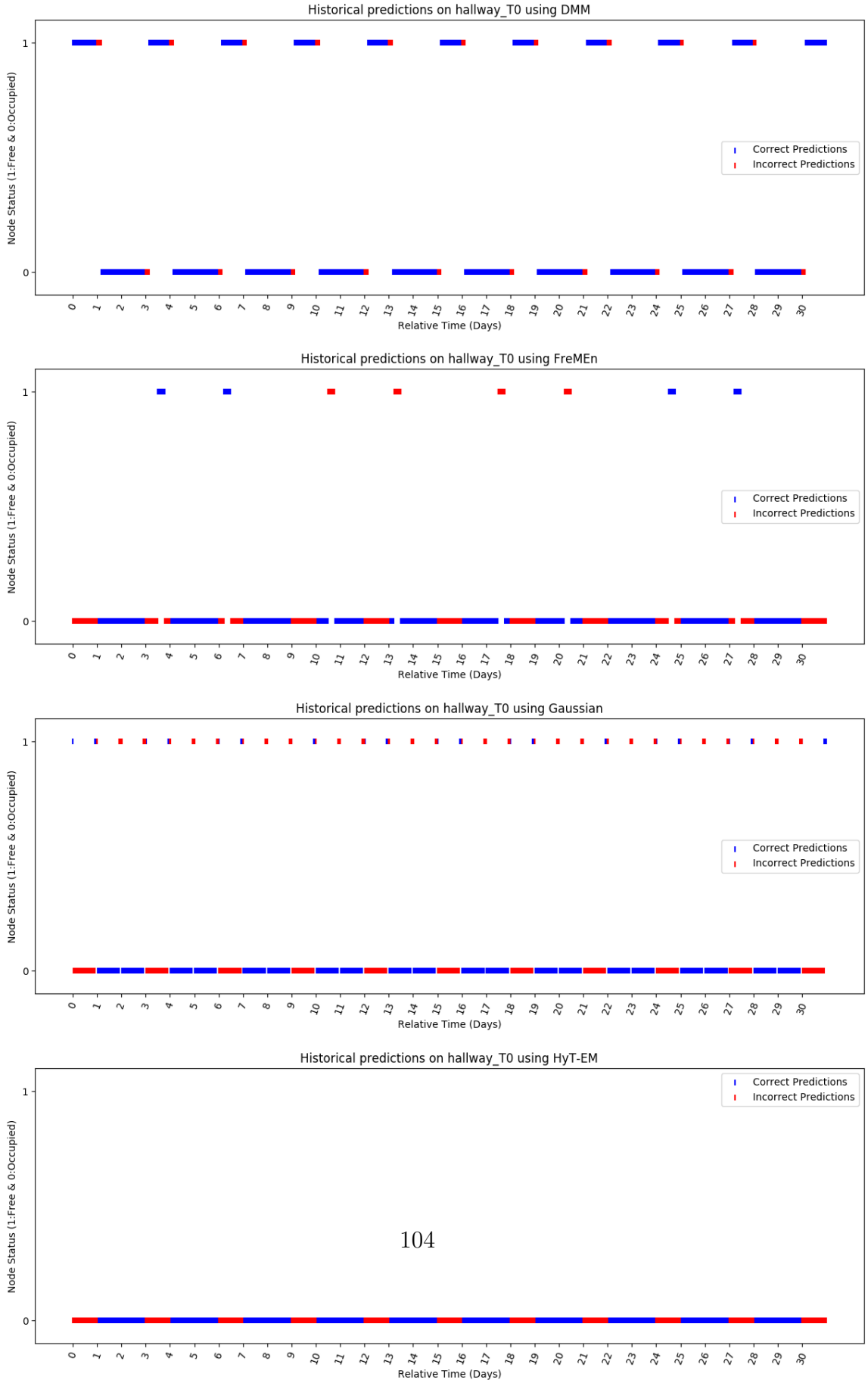

Figure A.13: Historical Recreations - Hallway Trash Section 0

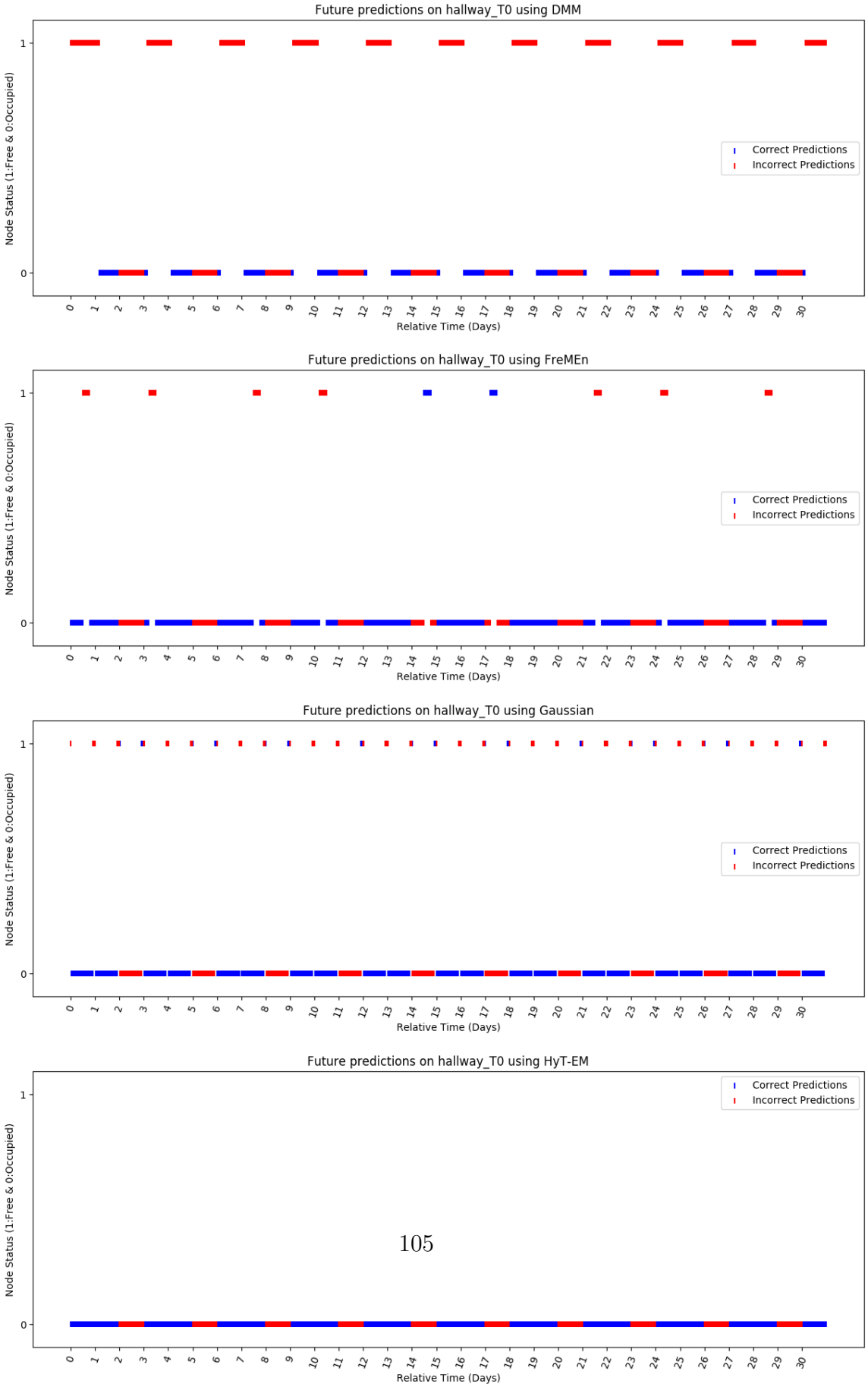

Figure A.14: Future Predictions - Hallway Trash Section 0

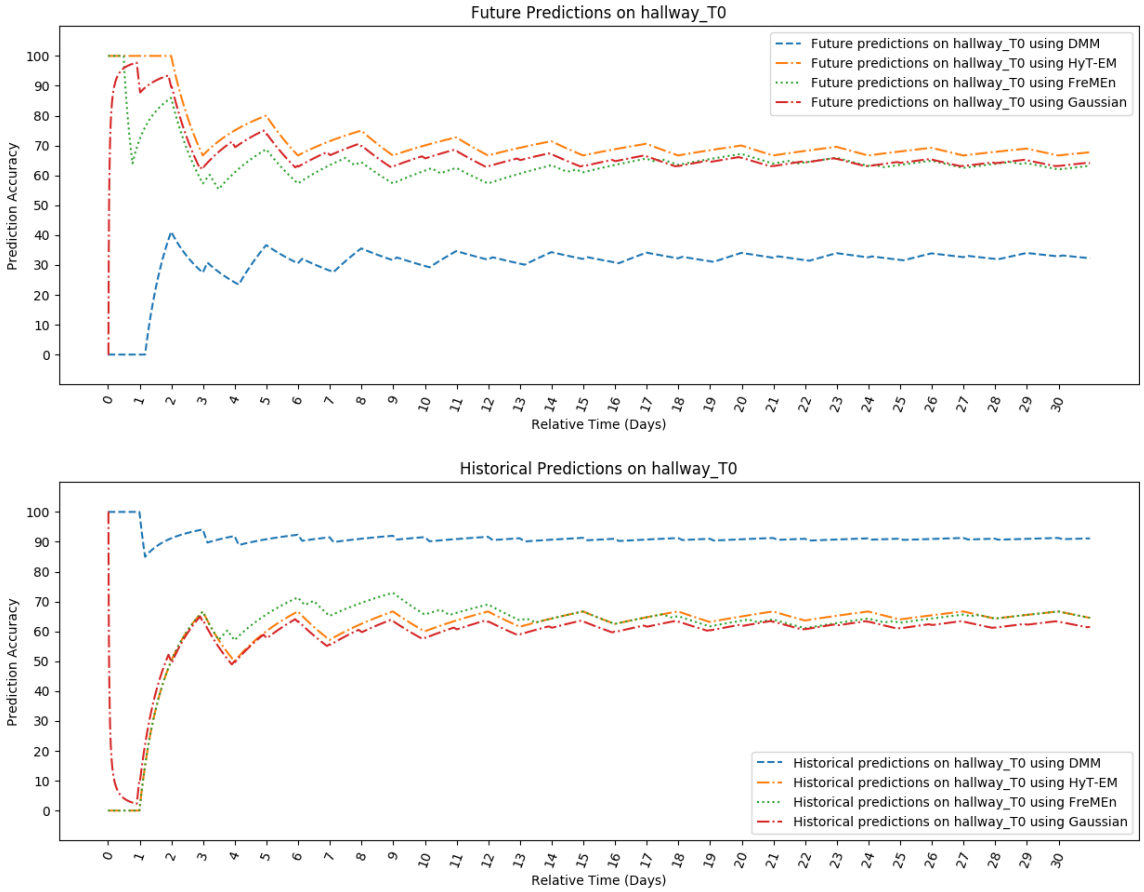

Figure A.15: Model Accuracy Over Time - Hallway Trash Section 0

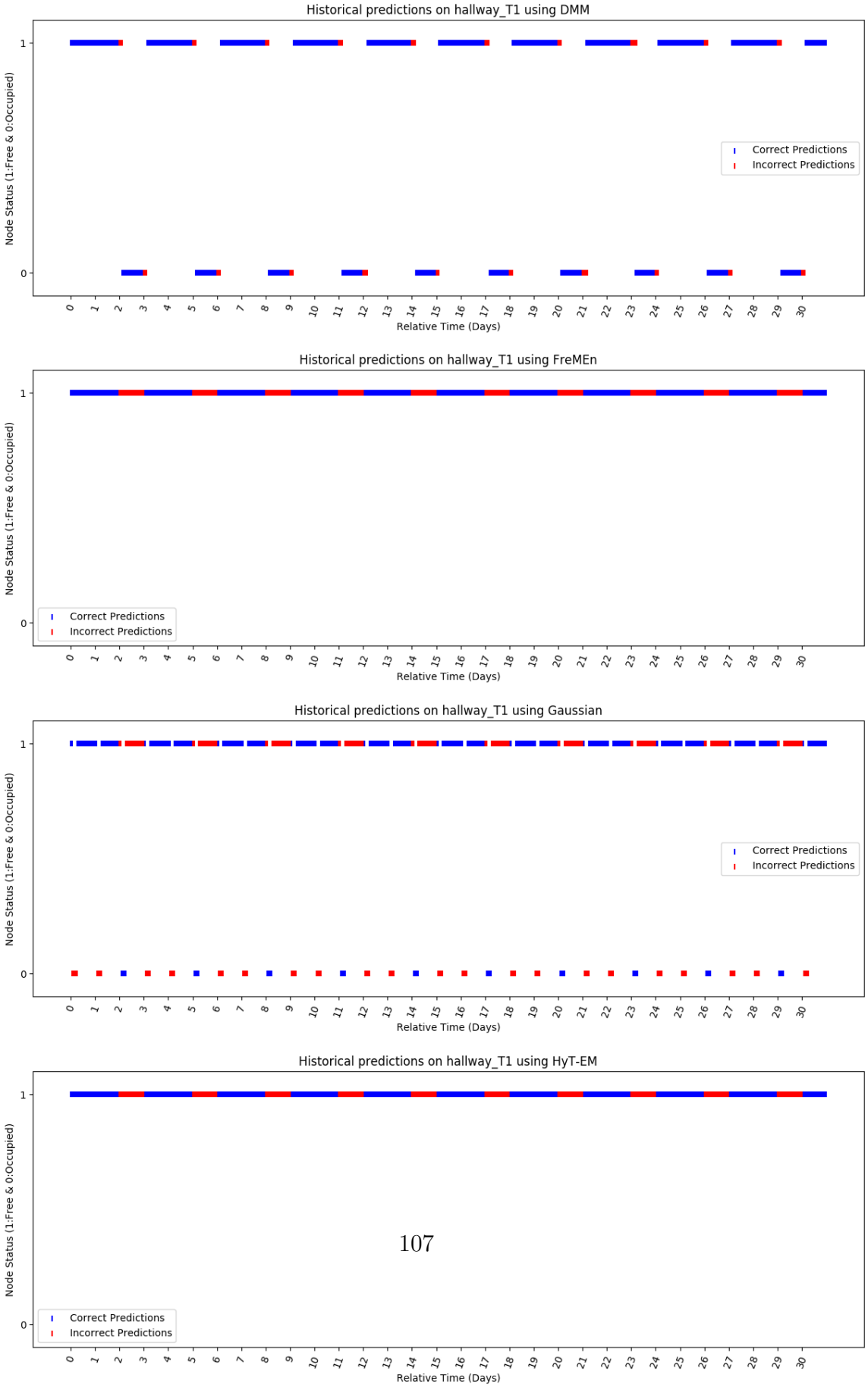

Figure A.16: Historical Recreations - Hallway Trash Section 1

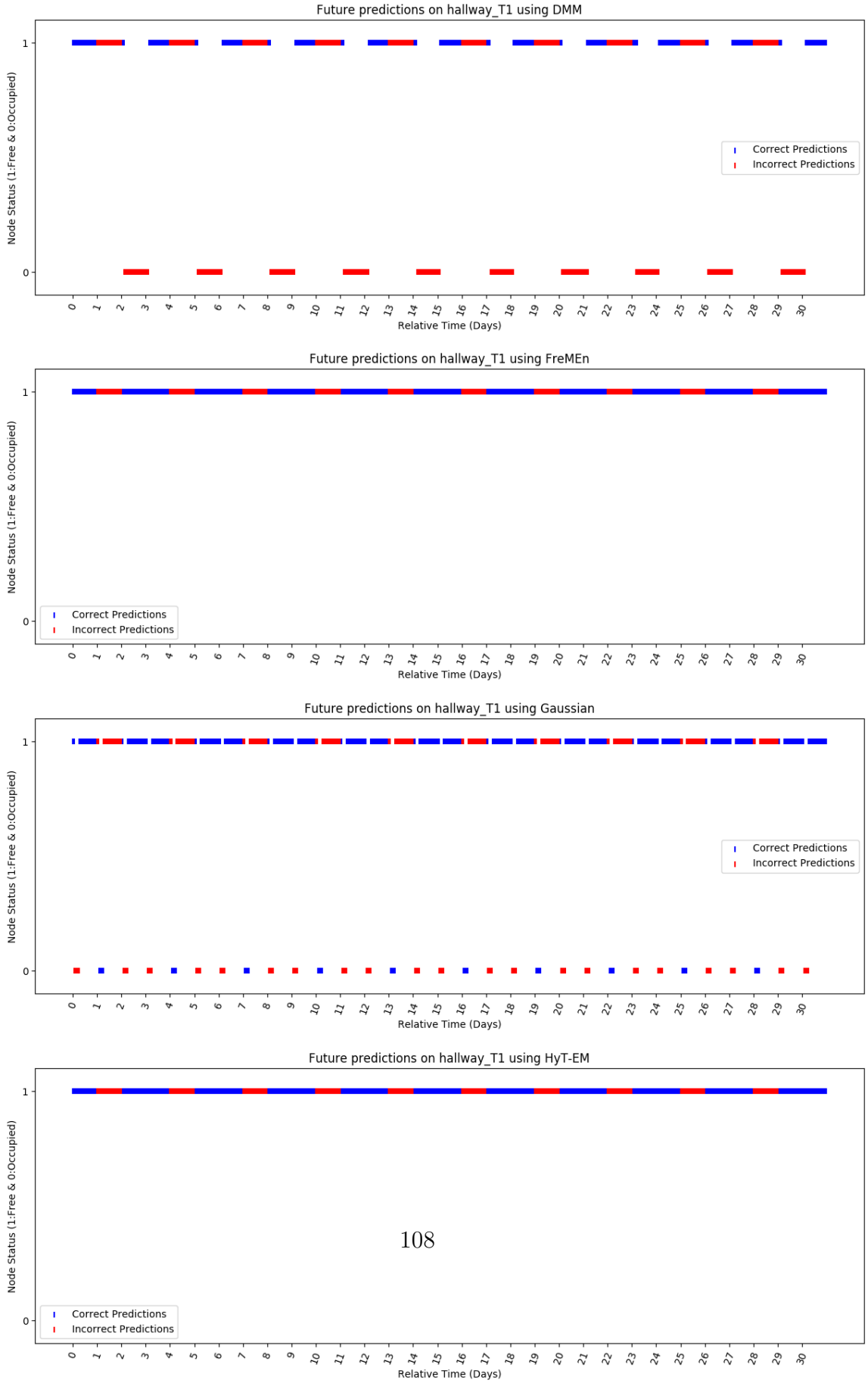

Figure A.17: Future Predictions - Hallway Trash Section 1

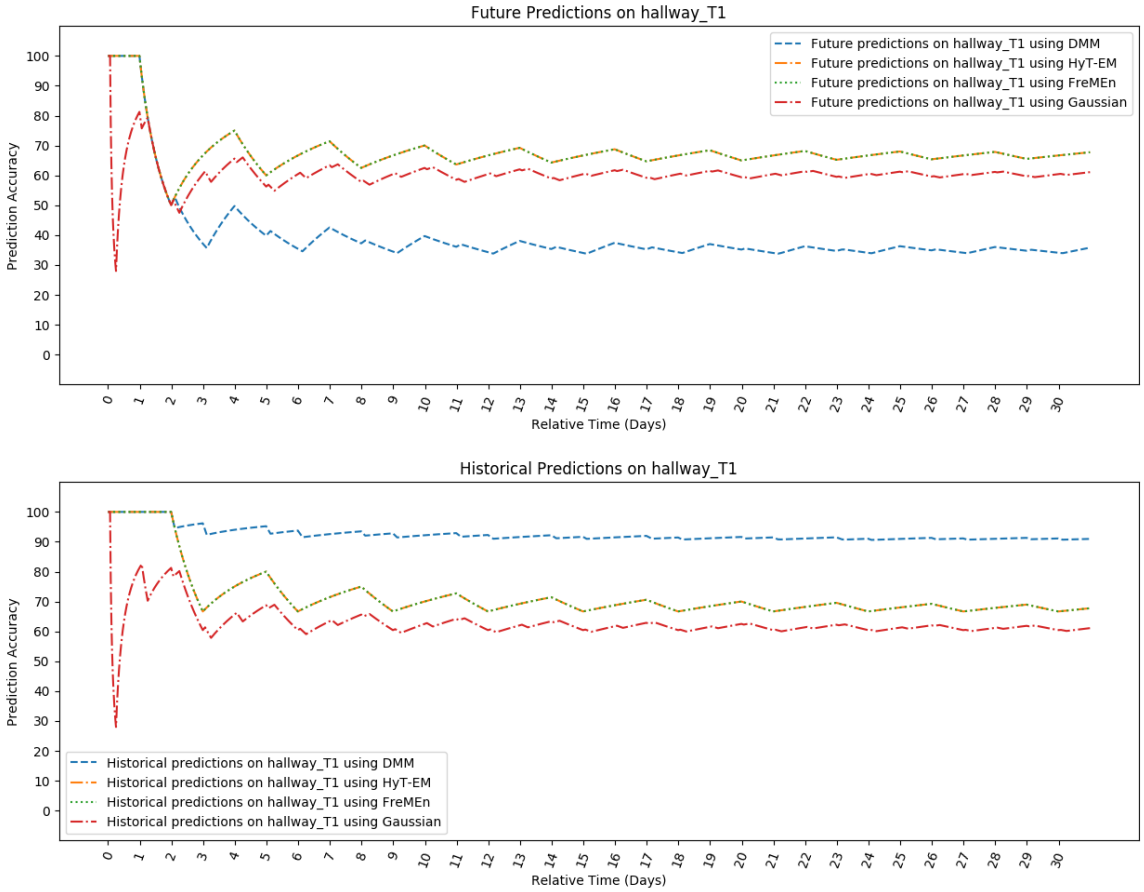

Figure A.18: Model Accuracy Over Time - Hallway Trash Section 1

B

High Resolution Elevator Experiment Results

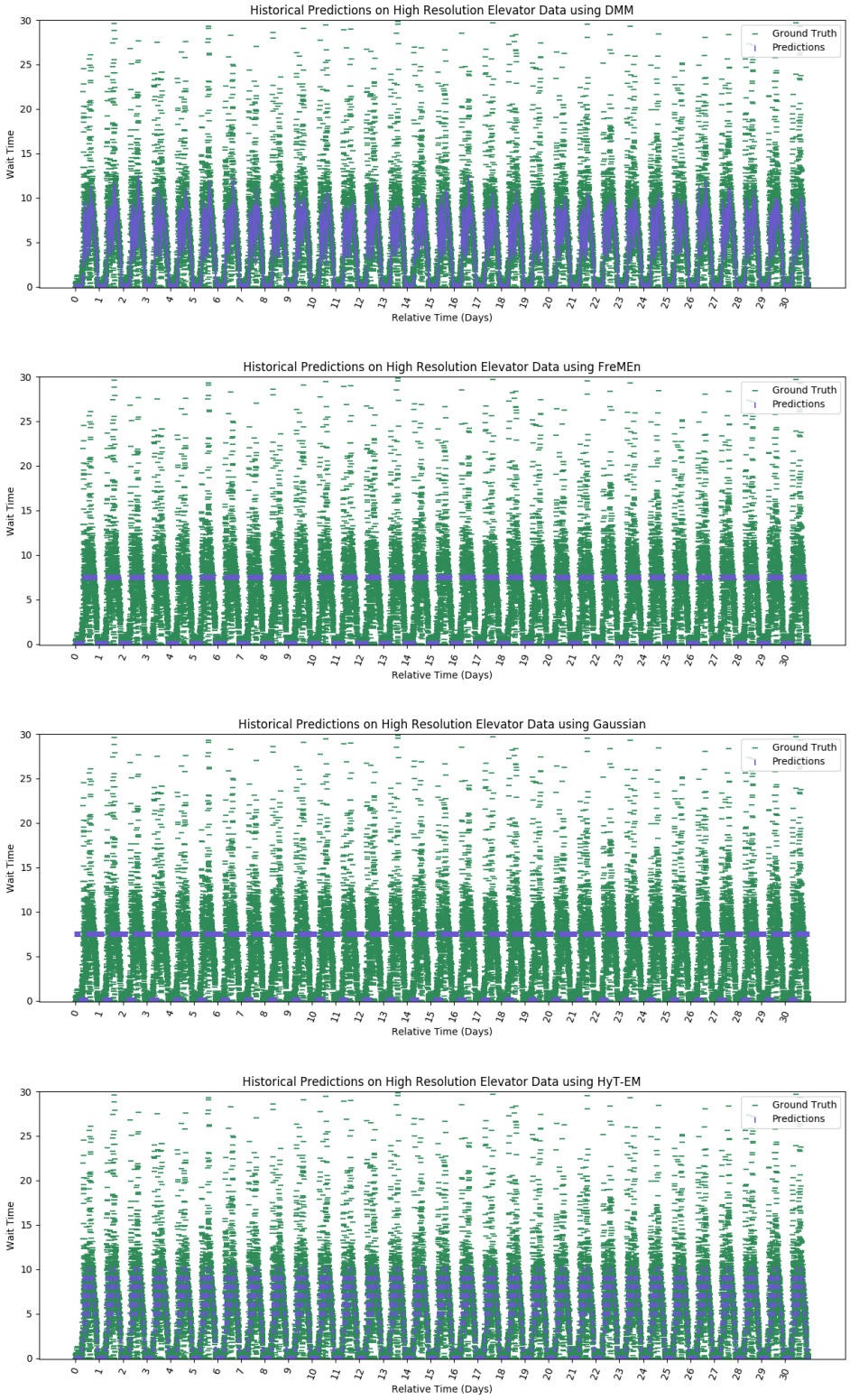

Figure B.1: Historical Recreations - High Resolution Elevator Data

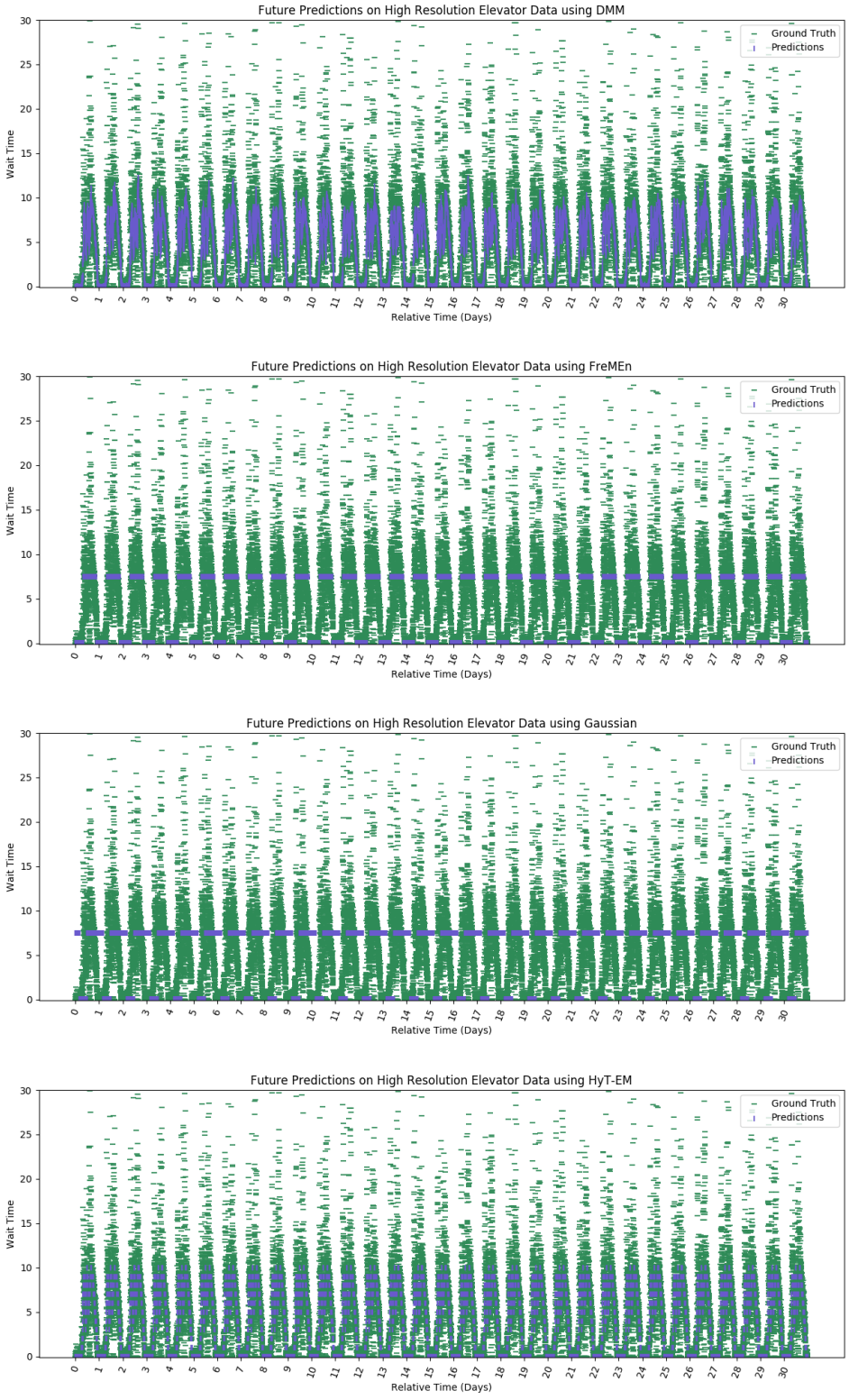

Figure B.2: Future Predictions - High Resolution Elevator Data

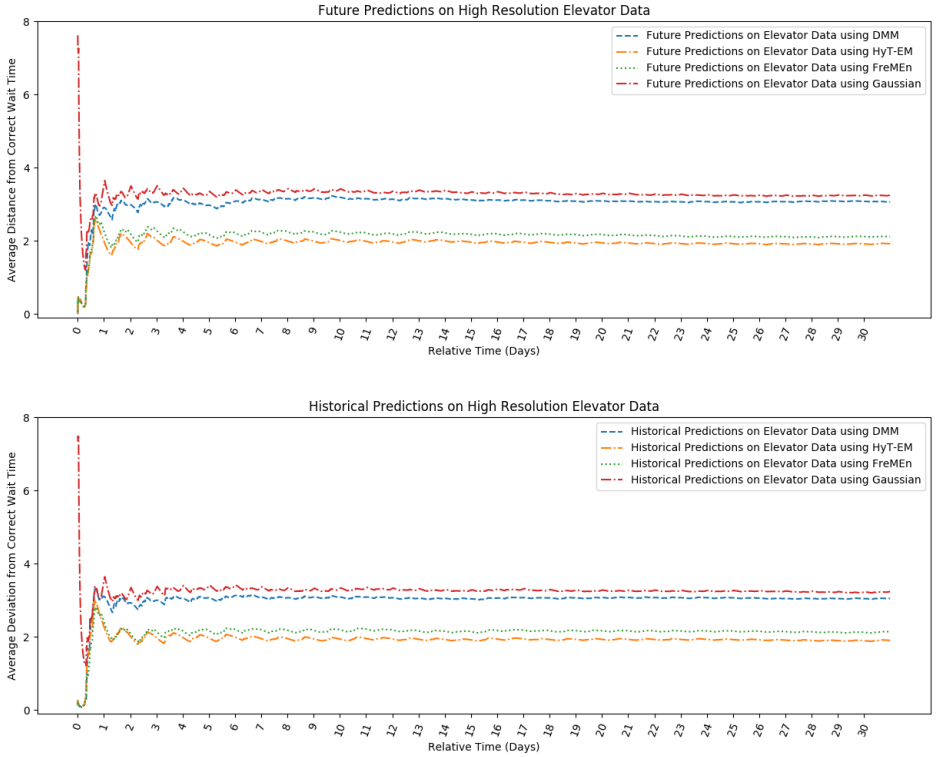

Figure B.3: Model Accuracy Over Time - High Resolution Elevator Data

References

[1] A. Elfes. A sonar-based mapping and navigation system. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 3:19–24, 1985.

[2] C. Coppola, and T. Krajník, and T. Duckett, and N. Bellotto. Learning Temporal Context for Activity Recognition. *European Conf. On Artifical Intelligence (ECAI)*, 285:107–115, 2016.

[3] D. Arbuckle, and A. Howard, and M. Mataric. Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 1:409–414, 2002.

[4] D. Meyer-Delius, and M. Beinhofer, and W. Burgard. Occupancy Grid Models for Robot Mapping in Changing Environments. *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, pages 2024–2030, 2012.

[5] F. Jovan, and J. Wyatt, and N. Hawes, and T. Krajník. A poisson-spectral model for modelling temporal patterns in human data observed by a robot. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016-November:4013–4018, 2016.

[6] F. Ramos, and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *Intl. Journal of Robotics Research (IJRR)*, 35: 1717–1730, 2016.

[7] J.M. Santos, and T. Krajník, and J.P. Fentanes, and T. Duckett. Lifelong Information-Driven Exploration to Complete and Refine 4-D Spatio-Temporal Maps. *IEEE Robotics and Automation Letters (RA-L)*, 1(2):684–691, 2016.

[8] J.P. Fentanes, and G. Cielniak, and C. Dondrup, and T. Duckett. Spectral Analysis for Long-Term Robotic Mapping. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3706–3711, 2014.

[9] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[10] M. Siff. Computational complexity, 1997. URL `http://pages.cs.wisc.edu/~siff/CS367/Notes/complexity.html`.

[11] N. Hawes, et. al. The STRANDS Project: Long-Term Autonomy in Everyday Environments. *IEEE Robotics and Automation Magazine (RAM)*, 24(3):146–156, 2017.

[12] R. Senanayake, and L. Ott, and S. O'Callaghan, and F. Ramos. Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments. *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 3918–3926, 2016.

[13] R. Senanayake, and S. O'Callaghan, and F. Ramos. Learning highly dynamic environments with stochastic variational inference. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2532–2539, 2017.

[14] R.M. Karp. On-Line Algorithms Versus Off-Line Algorithms: How Much is it Worth to Know the Future. *Proc. of Information Processing Congress (IFIP)*, pages 416–429, 1992.

[15] T. Duckett, and P. Biber. Dynamic Maps for Long-Term Operation of Mobile Service Robots. *Proc. of the Robotics: Science & Systems Conf.* , pages 17–24, 2005.

[16] T. Krajník, and J.P. Fentanes, and O.M. Mozos, and T. Duckett, and J. Ekekrantz, and M. Hanheide. Long-term topological localisation for service robots in dynamic environments using spectral maps. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4537–4542, 2014. URL `http://dx.doi.org/10.1109/IROS.2014.6943205`.

[17] T. Krajník, and T. Vintr, and S. Molina, and J.P. Fentanes, and G. Cielniak, and T. Duckett. Warped Hypertime Representations for Long-term Autonomy of Mobile Robots. 2018. URL `http://arxiv.org/abs/1810.04285`.

[18] T. Krajník, J.P. Fentanes, J.M. Santos, and T. Duckett. FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments. *IEEE Trans. on Robotics and Automation*, 33(4):964–977, 2015.

[19] T. Kucner, and J. Saarinen, and M. Magnusson, and A.J. Lilienthal. Conditional transition maps: Learning motion patterns in dynamic environments. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1196–1201, 2013.

[20] Z. Wang, and P. Jensfelt, and J. Folkesson. Modeling Spatial-Temporal Dynamics of Human Movements for Predicting Future Trajectories. *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2015.