Understanding Drill Data for Autonomous Application

Sarah Boelter
University of Minnesota Twin-Cities
Minneapolis, United States
boelt072@umn.edu

Ebasa Temesgen
University of Minnesota Twin-Cities
Minneapolis, United States
temes021@umn.edu

Brian J. Glass
NASA Ames Research Center
Moffett Field, United States
brian.glass@nasa.gov

Maria Gini
University of Minnesota Twin-Cities
Minneapolis, United States
gini@umn.edu

ABSTRACT

In high-risk, high-cost environments like Mars it is necessary for robotic agents to either respond to or reason through potential indicators of trouble before they escalate to problems that effectively mean mission failure. Currently, no broadly applicable solution exists to give a complicated and specialized agent like The Regolith and Ice Drill for Exploring New Terrain (TRIDENT) the ability to understand the rate of its progress on a task or the situational awareness to know when a situation might escalate to a drilling fault. We examined logged data from previous field experiences to better understand potential drilling faults TRIDENT will have to reason through. We applied time series analysis techniques to determine what trends in the data exist during faults and if change point analysis or other machine learning methods could be used to predict faults of the drill.

KEYWORDS

Space exploration, autonomy, planetary drilling and sampling, time series data

ACM Reference Format:
Sarah Boelter, Ebasa Temesgen, Brian J. Glass, and Maria Gini. 2024. Understanding Drill Data for Autonomous Application. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024,* IFAAMAS, 9 pages.

1 INTRODUCTION

The Atacama Rover Astrobiology Drilling Studies (ARADS) project was developed to explore and understand the mobility and distribution of salts, compounds, biosignatures, and extant life down to a 1 meter depth in planetary environments using a rover with an attached drill [7]. The rover contains other analysis equipment like a sample transfer arm, soil chemistry analyzer, and other mechanisms not directly affecting the drill or its software. The rover provides mobility and power to the drill and directly impacts its function. This entire system including the drill and rover was tested in Chile in September 2019. Diagnostic and automation software, including onboard planning and scheduling and fault diagnosis and recovery, was prototyped [10].

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). This work is licenced under the Creative Commons Attribution 4.0 International (CC-BY 4.0) licence.

Part of the onboard planning and scheduling software, Icebreaker Executive (IbExec), coordinates the control of multiple robotic subsystems and ARADS missions. Stucky et. al. [26] discuss its capabilities including commanding a robotic arm and robotic drill in coordination with one another. At the core of IbExec is the Plan Execution and Interchange Language (PLEXIL), a synchronous reactive language used to write procedures and that enables the extensible hardware interface. Dowek. et. al. [5] describe how PLEXIL was developed by NASA to support autonomous commanding and monitoring for a variety of space systems that have limited computational resources. PLEXIL programs, called plans, specify actions to be performed by an executive system. A PLEXIL plan consists of nodes organized in a tree-like structure that can run simultaneously and monitor conditions in an environment.

The primary ARADS component we focus on is the drill, known as "The Regolith and Ice Drill for Exploring New Terrain" (TRIDENT), shown in Figure 1. It is a 1-meter rotary percussive drill manufactured by Honeybee Robotics based on nearly a decade of development. This drill is designed to generate cuttings from a borehole for analysis by other ARADS system components. TRIDENT is the baseline for the Volatiles Investigating Polar Exploration Rover (VIPER) mission set to launch at the end of 2024 [29] [10]. ARADS uses IbExec as its onboard autonomy software to control payloads, such as TRIDENT. IbExec has helped automate tasks and eliminate user error, however, it still requires constant user oversight and assessment of the environment to select which tasks to do. Glass et al. [12] detail the problems of planetary drilling and why making it autonomous is not straightforward. In summary, it is impossible to know without prior geological surveying what range of rock types exist below the surface and what will be encountered. It will be difficult to have a one-size-fits-all program to all scenarios, since drilling techniques differ with changes in target composition, in addition to external factors like temperature. We spent time field testing the TRIDENT drill with NASA Ames and the Goddard Institute Field Team (GIFT) in the Bishop Tuff in Fall 2023 to understand how TRIDENT works in the field. It was observed that TRIDENT gives indicators, visually and otherwise, of a catastrophic failure well before it happens. Data graphed from collected log data indicate that a catastrophic failure has a high likelihood of being predicted before it happens, but the necessary components to intelligently monitor and execute commands to avoid failure do not exist yet. Instead of using a more traditionally proposed route to make TRIDENT and its systems fully autonomous to operate without human input, we propose predicting situational risk factors for critical errors for the TRIDENT drill based on collected test data and developing a planning model for TRIDENT to predict when human intervention may become necessary. This will enable TRIDENT to enter into a stable state until limited human oversight can assist with decision making.

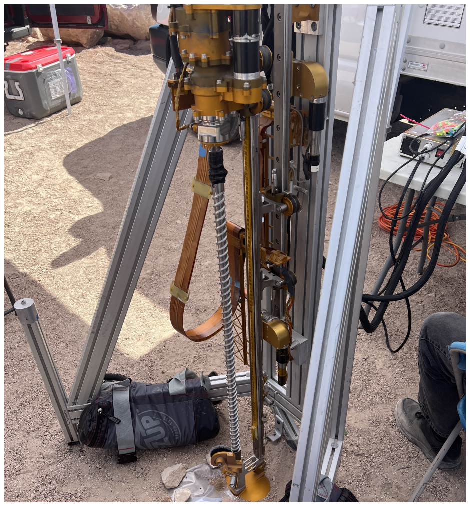

Figure 1: TRIDENT drill during field testing in Bishop, California in Summer 2023

This paper focuses on the analysis of collected field data and its impacts on future model development. The primary focus is on identifying and extracting usable data, understanding what faults exist within it, identifying trends between different variables using time series analysis, and discussing their impact on further work.

2 RELATED WORK

Autonomy under high-risk and little oversight conditions is not a new concept and many ideas have been proposed on how to best solve this for decades. Muscettola et al. [17] discuss issues surrounding the Sojourner rover and its mission to Mars in 1997. Since the rover lacked onboard autonomy software, operating it for several months was incredibly taxing on the ground crew, substantiating the need for onboard autonomy with minimal human oversight in future missions. They described three requirements for space operations: autonomy for long duration, methods with guaranteed success, and high reliability. Even though issues on autonomy have existed for decades, we still do not have an out-of-the-box solution. Instead, Luckcuck et al. [15] discuss focus areas to solve many of the issues brought up in [17]. The suggestions include modeling and simulation of the physical environments, identification of hazardous situations, and formal verification of robot systems.

In addition to developing a theoretic solution to the identified autonomy problem using Artificial Intelligence, we must assume we will have to implement our solution on hardware at some point in the process. Utilization of PLEXIL for a large portion of our solution is expected, so we need to address any necessary considerations. The PLEXIL [5] software framework allows for adaptability; its adapter module allows for the swapping of communication methods without the need to alter control scripts. For example, before TRIDENT, the same PLEXIL scripts were used to control two of ARADS's predecessor drills, LITA and CRUX, despite all three drills utilizing entirely different communication middleware. The adapter module also enables a multitude of independent systems to be brought together under the same control umbrella, which means PLEXIL can act as a central control executive for an entire mission [26]. Additionally, numerous tools exist to assist or provide a framework for model checking, particularly PLEXIL5 [19] and LTSA [3].

Work surrounding drilling automation with TRIDENT is not new either. Glass et al. [12] discussed the need for automation with Drilling Automation for Mars Exploration (DAME) in 2005. The project focused on refining drill design and considering potential ideas for automation. These ideas were further discussed by Glass et al. [8] with the DAME, MARTE, and CRUX drills and how the Icebreaker drill imported some prototype automation software from DAME, but a rework was necessary since it did not work on flight computers. These ideas influenced what would become IbExec. Additional drilling work was conducted on Mars [16, 22].

Time series data poses more potential complications than traditional data sets for modeling and forecasting. This is due to the typically high complexity large scale of data sets and difficulties with forecasting. Liu et al. [14] discuss forecast methods for time series data. Artificial Neural Networks (ANN) are typically used for time series forecasting, however they tend to over-fit for small data sets, which would be problematic since TRIDENT drill data sets can be as little as several seconds. Gaussian Process Regression is a good candidate for this data set since it requires fewer parameters, performs generally better than ANNs, and performs better with smaller data sets. However, it has a high computational cost which would be amplified with the varied sizes of our data sets.

In addition to forecasting for time series data, we also utilize offline Change Point Detection (CPD) algorithms to detect shifts in the mean. With our use of offline methods, it is worth noting online algorithms that may fit our future use case. Notably, Adams et al. [1] discussed Bayesian CPD and its applicability to robotics. Specifically, how robots must navigate using past sensor data in an environment that may have abruptly changed. Due to the changing and unknown planetary environment TRIDENT may navigate, online CPD methods like Bayesian could be an excellent fit for later objectives, assuming offline methods are sensitive to our current fault datasets.

3 PRELIMINARIES

To create a formal problem definition, we need to understand what relationships exist in the data collected. Figure 2 is a sliced representation of the types of drill data collected and how they can be annotated and represented to be understandable at a casual glance. The figure in particular shows a hard-material and corkscrewing fault with a predecessor drill during testing in Antarctica in 2013. Some of the most pertinent data logged from the drill and its immediate components are shown in the graph and include variables related to the specific position of the drill bit in space ($zPosition$, $augerPosition$), forces on and applied to the bit and motor ($WobAdjusted$, $AugerPower$, $ZVelocity$, $PercPower$), and variables related to the immediate external environment ($HoleDepth$,

$BitTemperature$). In addition to corkscrewing faults and hard materials faults, these time series data points can give us indications of other faults like choking and binding faults. Choking faults occur when excess drill cuttings accumulate in the bottom of the hole with nowhere to go, and binding faults occur when cuttings get stuck between the bit and the side of the hole. The data is non-static over time, and depending on conditions and situations could span as little as a few seconds to several minutes.

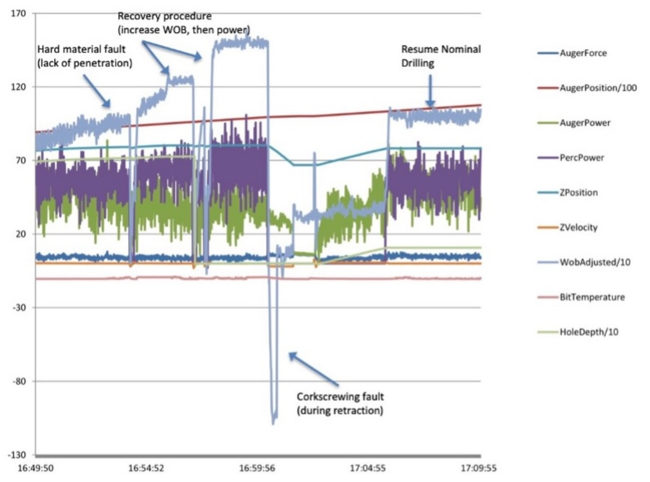

Figure 2: Data collected while drilling in Antarctica in 2013 showing a hard material fault and then a subsequent corkscrewing fault for automated drilling [9]

Once TRIDENT's data relationships have been established, we will use AI planning to understand drill progression based on environmental feedback from sensors. Since the environment of TRIDENT's potential errors is complex, we can approach this problem as a sequential decision problem. IbExec's modules can be represented by several states described by the terms calibrate, home, deploy, drill, retract, recovery, and stow, as shown in Figure 3.

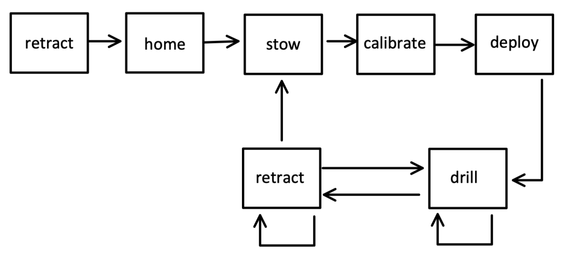

Figure 3: TRIDENT's behavior represented in several states described by the terms calibrate, home, deploy, drill, retract, recovery, and stow

The states of TRIDENT are: $S(Trident)$ = {*home, retract, stow, calibrate, deploy, drill*}. Each state in $S(Trident)$ has a set with all possible sub-states. For example, if $drill \subset S(Trident)$ the state drill could be represented by $S(drill)$ = {*0 mm, 5 mm, 10 mm, 15 mm, 20 mm, 25 mm, 30 mm, 35 mm, Retract, Early-Retract*}, with the initial state being 0 mm, and where each state in the set $S(drill)$ represents progress in drilling toward a specified depth in increments. Both terminal states are modeled as Retract, where TRIDENT retracts once the specified depth has been reached, or Early-Retract, where retraction is done before reaching the specified depth. In this example, a drilling depth of 35 mm is specified, and each increment of 5 mm is a state. For this example, we represent the states in a state map format as in Figure 4.

We can now define a Markov Decision Process (MDP). We can represent our MDP for $drill \subset S(Trident)$ as a five-tuple $\langle S, A, T, R, \gamma \rangle$ where S is a set of all possible states reachable in $S(drill)$, A is a set of all possible actions in $S(drill)$, which for this example is defined as $A(Drill) = \{continue, retract\}$, where *continue* equates with continuing to drill, and *retract* equates to retracting before drilling to the specified depth. T is the transition model, which represents for each state s and action a the probability of reaching another state s'. We write T as $T(s'|s,a)$. In our current example, a transition probability could be $T(5mm|0mm, continue)$. R is defined as a reward function $R(s,a)$, where R represents the reward for taking action a in state s. For example, if we were at 0 mm and about to begin drilling, the reward function for taking the action *continue* would be $R(0mm, continue)$. γ is a discount factor where $0 < \gamma < 1$. The discount factor is used to favor a more immediate reward in the next state versus a more distant reward several states in the future. A solution specifies what the agent should do in any state that the agent might reach. A solution is called a policy. π is a

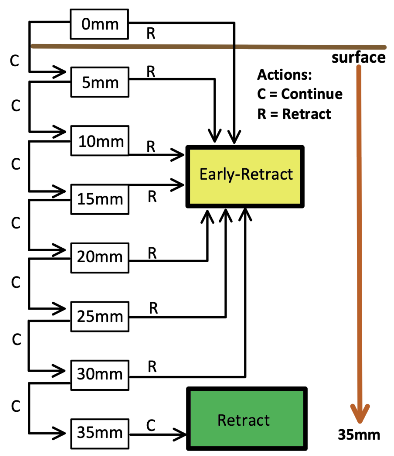

Figure 4: Possible states for drilling depth 35mm in S(Drill).

policy, and $\pi(s)$ is the action recommended by the policy for state s. The optimal policy, which produces the highest expected utility, is denoted by $\pi*$.

We can solve this problem with the value iteration algorithm which uses the Bellman equation:

$$U(s) = \max_{a \in A} \left\{ \sum_{s'} T(s'|s,a) \left[R(s,a) + \gamma U(s') \right] \right\} \tag{1}$$

where $U(s)$ represents the utility of state s, which is a measure of the desirability of a state. The set A encompasses all possible actions that can be taken. The transition probability, $T(s'|s,a)$, quantifies the likelihood of moving from state s to state s' using action a. The reward for taking action a in state s is given by $R(s,a)$. γ is the discount factor, indicating the relative importance of future rewards compared to immediate rewards. The expected utility for executing a policy is the expectation of the weighted sum of the rewards. The value iteration algorithm calculates the utility of each state and uses the utilities to select an optimal action in each state.

To influence online decision-making, we will create utility and reward probability functions and represent risk and reward using online data from PLEXIL nodes. Since PLEXIL nodes can run concurrently,we can monitor several key data related to position and progression, forces applied to the bit and motor, and environmental variables at once. When a node fails, or returns a changed value, it can trigger an action within our model that brings us closer to either terminal state. We will define our utility function(s) thresholds from previously analyzed combinations of data that often predict certain errors. This will help define what actions are taken in the model. When the model reaches a transition state with a high enough probability fault, the reward function will influence actions that lead to the second terminal state, which may lead to TRIDENT backtracking, entering a stable recovery mode, or doing another action depending on the defined action. The ultimate goal is to make progress but to evaluate values for risk vs reward to inform an optimal policy to avoid a mission-critical failure.

4 METHODS

4.1 Experimental Setup

Before analysis of the drill data, it is important to understand the data types and how data must be processed. We analyzed data collected at field events over the last decade to determine what data would be the best candidates for processing initially. We examined both electronic data and hand written notes from field testing. Most of the time-series data has a similar length, however, there were some examples both exceptionally long or short, potentially due to different types of material being drilled into.

We considered all possible instances where data was collected. Prior to 2016, the CRUX drill was used [8], which had different hardware than the TRIDENT drill. Data from CRUX were recorded in data space-separated values (SSV) plain text files. After 2016, TRIDENT was used and data were recorded in Hierarchical Data Format version 5 (HDF5) binary files, which require more work to process. We could not consider any data prior to 2016 due to the method of data storage. Additionally, the drill hardware used for the experiment, CRUX, had different calibrations and settings which would potentially cause variations if compared side by side with more recent data. We combed through data collected after 2016 and notes in search of data that resembled choking and binding faults. We were able to collect two visually clean instances of choking and faults from the Haughton Mars Project (HMP) at Devon Island, Canada [11] and subsequently at the Goddard Instrument Field Team's September field deployment to the tuff deposits near Bishop, CA in 2023 [25]. We also collected eleven instances of clean choking and binding faults from the Life-detection Mars Analog Project (LMAP) in Rio Tinto, Spain in 2017 [21].

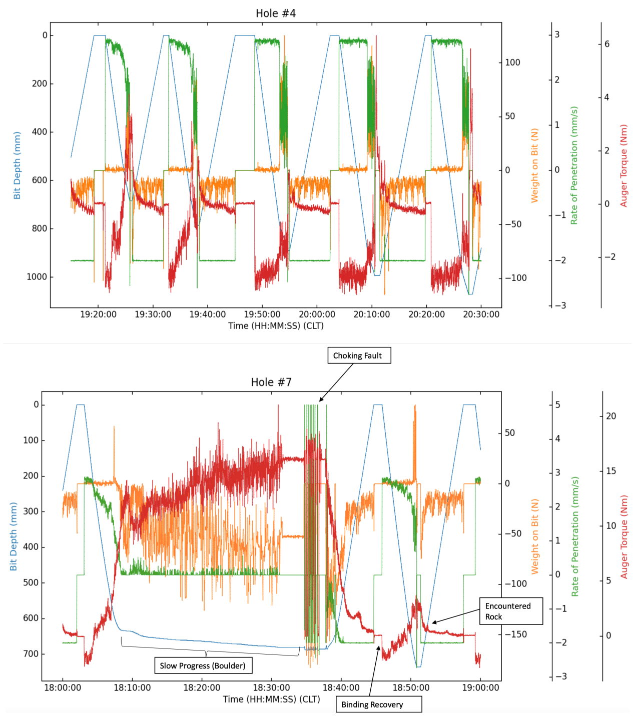

Figure 5: Upper image shows normal drilling operations during testing. Bottom image illustrates choking fault during testing. Both testing done in Devon Island, Canada [11]

Figure 5 shows two examples of time series data for two different holes on a recent field exercise. The first illustrates normal drilling operation and the second illustrates a choking fault. From observing plots with field notes, we inferred that choking and binding faults tend to raise Torque and Weight on Bit while the Rate of Penetration is roughly zero. This indicates that multiple variables could be used to predict faults, however, we decided to examine variables individually. We first decided to further examine Torque data in our time series analysis due to its distinct and similar shape across all fault graphs, and later focus on other data features like Rate of Penetration and Weight on Bit.

4.2 Time Series Analysis and Predictions

Time series analysis plays a crucial role in understanding the patterns and predicting future trends in the data. This section focuses on applying time series analysis techniques to analyze the drill's performance of data of varying lengths in a similar domain and identifying similarities between faults. Then forecasting will be utilized to examine whether trends can be predicted with machine learning techniques like Support Vector Machines (SVM), Random Forest, AdaBoost, and Long Short Term Memory (LSTM) networks.

Dynamic Time Warping. A large concern with TRIDENT data is the availability of fault data and the usability of previously collected data. Most datasets of faults are similar in length, but some are either very short or very long in duration, and it is difficult to compare data of different lengths. Collecting data requires assembling the standalone TRIDENT drill with its many components. When assembled, two people must attend to its operations and ensure detailed notes are being taken. While not necessarily required, traveling to sites with ideal materials that may not easily be simulated is helpful for testing for specific faults. Faults are not guaranteed during testing, hence we must make use of every data set collected. While most data are similar in duration, we sometimes find datasets of incredibly short duration or much longer duration than average. Comparing data outside the average range of duration side-by-side with data sets existing within it leads to inaccuracies. Using tools like Dynamic Time Warping (DTW) [20] we can help map data of unequal length to the same dimensional space. DTW can be represented by an $n \times m$ matrix, where n and m represent the length of two datasets x and y.

$$DTW(x, y) = min_{\pi} \sqrt{\sum_{(i,j) \in \pi} d(x_i y_j)^2} \tag{2}$$

DTW finds the squared Euclidean distance between each point in each graph, and tracks the most cost-effective path in terms of distance. In addition to DTW, DTW Barycenter Averaging (DBA) is a global averaging method for DTW [18]. Instead of an iterative averaging approach, DBA takes in multiple sets of data (D) and creates an average from multiple sets without creating bias to one piece of data over the other. We elected to try the DBA expectation-maximization approach [18] instead of the sub-gradient descent approach, which is generally more efficient for larger data sets [23].

$$min_{\mu} \sum_{x \in D} DTW(\mu, x)^2 \tag{3}$$

We used tslearn [27] package for python to see what initial results looked like. Below in our initial test, we can see the input of two data sets of different lengths where we applied DBA.

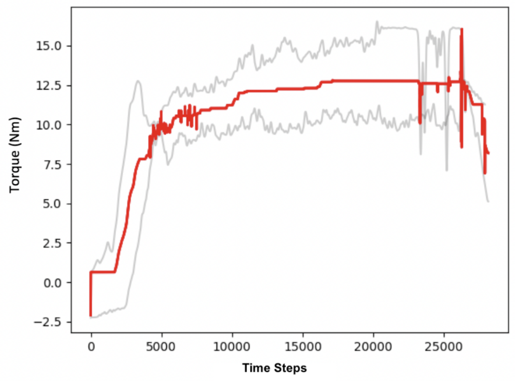

Figure 6: DTW and Barycenter Averaging used to find the average of two time series datasets of different lengths

Forecasting. In addition to DBA, we explored several forecasting methods to help us better understand if faults can be predicted and in what contexts. Once we have decided on the prediction method, we will use it to learn the probability of fault and how it will influence decision-making. In particular, we were interested in exploring how conventional machine learning and neural network methods could detect in our time series data sudden data changes, like a mean shift or an increase in variance before a fault, and if so, how many steps ahead. These will help inform us what online change prediction methodology to choose in later stages since early fault detection will require rapid decision-making.

Among the forecasting models evaluated, our analysis incorporated both traditional machine learning techniques and advanced neural network architectures, specifically:

- *Long Short Term Memory (LSTM)* networks were employed given the complexity and volume of the drilling data. LSTMs, which are a type of Recurrent Neural Network, are effective in capturing long-term dependencies in time series data.
- *Support Vector Machines (SVM)* were employed for their effectiveness in high-dimensional spaces and versatility in capturing complex data relationships through kernel functions.
- *Random Forest* was used for robustness and ability to model non-linear relationships through an ensemble of decision trees, providing insight into feature importance for prediction.
- *AdaBoost* was utilized to enhance the predictive performance by combining multiple weak learners into a strong ensemble model, aiming to improve fault detection accuracy.

Evaluating LSTM for Advanced Forecasting. The decision to emphasize Long Short-Term Memory (LSTM) networks in our study was due to their unique ability to prioritize recent observations in time series data, critical for early detection of faults. The LSTM model was meticulously configured and trained to forecast up to 100 points ahead, providing a significant lead time for fault detection and prevention.

This LSTM model employs a structured architecture to effectively capture long-term dependencies inherent in the dataset. The network is composed of an input layer tailored to accept a single feature (input dimension of 1), followed by two stacked LSTM layers with 32 hidden units each to enhance the model's ability to learn complex temporal patterns. To connect the recurrent layers to the output, a linear layer transforms the final hidden state's output to the desired prediction length of 100, enabling the model to output a sequence of predictions based on the input sequence. The model utilizes the Adam optimizer with a learning rate of 0.01, leveraging its adaptive learning rate properties for efficient convergence.

Table 1: Evaluation metrics for forecasting models of fault data

Model	R-squared (R2)	MAE	MSE
SVM	0.9339	0.5300	1.6615
Random Forest	0.9599	0.4998	1.0076
AdaBoost	0.9371	0.7439	1.5812
LSTM	0.9873	0.2309	0.1880

The comparative analysis of forecasting models, summarized in Table 1, underscores the Long Short-Term Memory (LSTM) model's performance in time series forecasting. With an R-squared value of 0.9873 and the lowest Mean Squared Error (MSE) of 0.1880, the

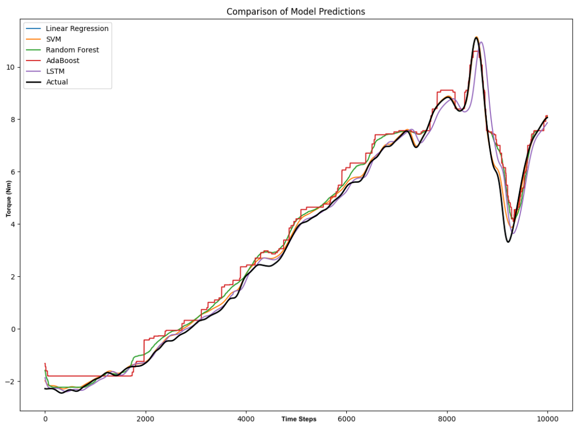

Figure 7: Comparative visualization of time-series predictions from the LSTM model alongside baseline models (Linear Regression, SVM, Random Forest, and AdaBoost).

LSTM model demonstrates a superior ability to accurately predict future data points and detect potential faults. This contrasts with traditional models like SVM, Random Forest, and AdaBoost, which, despite their robustness, exhibit higher MSEs, indicating a lower prediction accuracy. Figure 7 visualizes the time series predictions from the methods shown in Table 1. The results highlight the LSTM's potential to significantly enhance fault detection methods through advanced time series analysis, paving the way for its integration into predictive maintenance systems for drilling operations.

Predicting With Other Features. In the subsequent analysis, we looked into the predictive capabilities of the previously used models with additional features, specifically Weight on Bit and Rate of Penetration, to forecast drilling performance. The correlation analysis among Weight on Bit, Rate of Penetration, and Torque presents intriguing insights. While there is a relatively weak positive correlation between Weight on Bit and Rate of Penetration (0.069026), and a slight negative correlation between Weight on Bit and Torque (-0.211191), most notable is the strong negative correlation between Rate of Penetration and Torque (-0.810810). This suggests that as the rate of penetration increases, the torque required decreases, which could be indicative of underlying physical dynamics in drilling operations.

4.3 Change Point Detection

Because of the variety of data, manually encoding values in any update function would be tedious and potentially inaccurate if certain cases for behavior were not considered. According to Statham [24] in their thesis discussing planetary drilling, using online machine learning techniques could be too computationally complex for real-time detection of faults in a planetary environment. CPD algorithms could offer a less computationally complex solution for detecting sudden shifts of data in an online environment. We will explore CPD algorithms as a possibility for understanding data changes that differ significantly and suddenly from the mean or a steady rapidly increasing linear trend of time series data.

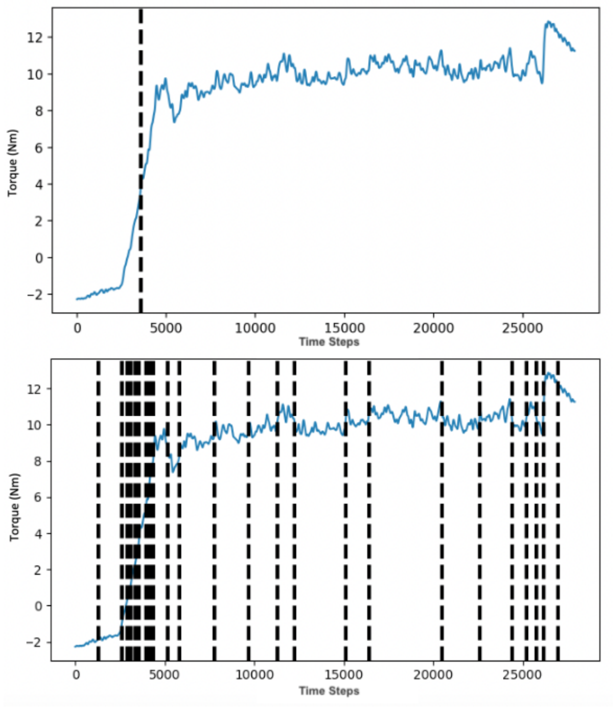

Figure 8: Offline CPD using the bottom-up approach. The top figure has a high ϵ specified leading to only large significant changes in the data being detected, while the bottom figure has a low ϵ specified, leading to more noise being detected as a change.

We explored several CPD algorithm options. We are still in an exploratory phase, so we are primarily concerned if CPD algorithms are appropriate for the types of data produced by TRIDENT. We focused on whether parameters for different offline CPD algorithms could be adjusted to fit our use case and successfully detect the observed change. We explored the offline Binary Segmentation [6] [2] and Bottom-Up approaches [13] [6]. Binary Segmentation works by dividing data into pieces based on change point locations. It has a low complexity and is usable whether the number of estimated change points is known in advance or not. Bottom-Up works by dividing up the data into sub-signals, and signals are merged depending on their similarity. We used the Python package ruptures [28] due to its simplicity and the l2-norm cost function.

We started with our time series Torque data, since they have the most observable sudden abrupt change compared to Weight on Bit and Rate of Penetration. With our raw data containing some noise and variations, we focused on adjusting the model to only catch the large significant changes from the mean. We did not specify the number of change points to detect. Both models allowed adjusting the residual norm, or the ϵ. With a small ϵ, we would catch every minor change from the local mean. With a larger ϵ we would decrease the sensitivity of the model to smaller changes in our dataset. As we increased our ϵ, fewer less-significant changes in the mean were caught and eventually only the major significant change was detected. This is demonstrated in Figure 8 using Bottom-up to detect change points in Torque data, where the top figure has a higher ϵ than the bottom figure.

We extended this methodology to our other data features. Unlike our Torque data, where we were looking for a single significant increase in the mean, in our Rate of Penetration data, we are looking for a sudden drop in the average mean as it approaches 0. The Rate of Penetration suddenly dropping to zero could indicate a potential fault when observed with Torque suddenly increasing. In Figure 9, we used both the Binary Segmentation and Bottom-Up approaches on Torque and the Rate of Penetration with similarly high ϵ. We then attempted to use these methods on Weight on Bit, which has a more subtle drop in mean value compared to the rate of penetration. Figure 10 shows the Weight on Bit mean changes detected by both Binary Segmentation and Bottom-Up approach.

5 RESULTS

Time Series Analysis and Predictions. We applied DBA to a few data samples and came out with an overall average equivalent to the shortest dataset. Since many of the datasets we encountered are similar in length, we concluded that this methodology would be best applied to datasets that were very long compared to the majority of time series data collected.

Table 2: Data Lengths for DTW and Averaging

Data Name	Length (Time steps)
HMP	28144
LMAP	27911
LMAP2	30456
Overall Average	27910

Upon training various models with these features, the performance metrics offer a comprehensive view of each model's forecasting accuracy. The SVM model yielded a respectable R-squared (R2) value of 0.7854, a Mean Absolute Error (MAE) of 7.1914 and a Mean Squared Error (MSE) of 139.2531. Conversely, the LSTM model exhibited a decline in performance under the same conditions with an R-squared value of -4.9957 and significantly higher error metrics (MAE: 57.0845, MSE: 3890.6091) over just ten epochs of training. The LSTM model struggled to adapt to this dataset's dynamics. These results underscore the SVM's ability in capturing the underlying patterns within this dataset.

Table 3: Comparison of forecasting models for 1-point and 100-point predictions of fault data

Model	1-point Prediction			100-point Prediction		
	MSE	R2	MAE	MSE	R2	MAE
SVM	1.5572	0.9386	0.4932	1.6615	0.9339	0.5300
Random Forest	0.7787	0.9693	0.4024	0.9837	0.9609	0.4972
AdaBoost	1.3035	0.9486	0.6578	1.5298	0.9391	0.7046
LSTM	0.0086	0.9994	0.0696	0.1499	0.9899	0.2144

In the analysis presented in Table 3, the LSTM model exhibits a notable performance advantage in forecasting torque values up to 100 steps ahead, achieving an MSE of 0.1499 and an R-squared value of 0.9899. This contrasts with the performance of traditional models such as SVM, Random Forest, and AdaBoost, which, while robust, show higher MSEs, indicating a lower prediction accuracy over both short and long-term forecasts in Figure 9.

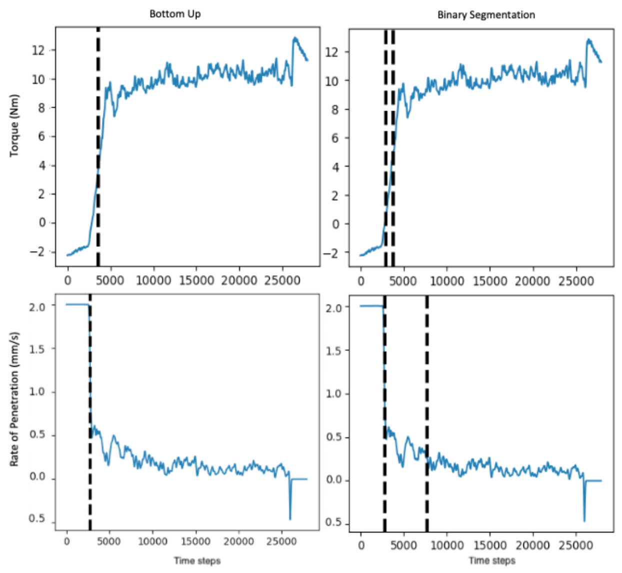

Figure 9: Results of CPD on Torque and rate of penetration time series data using Bottom-Up and Binary Segmentation.

Specifically, the LSTM model's precision in long-term predictions far exceeds that of its counterparts, with a significantly lower MSE and higher R-squared value, highlighting its efficiency in capturing temporal dependencies in time series data. This analysis emphasizes the LSTM's potential for detailed time series analysis, crucial for applications requiring advanced forecasting capabilities.

Change Point Detection. At first glance, both the Binary Segmentation and Bottom-Up approaches were able to detect the desired change point when ϵ was increased enough to make both models less sensitive to noise. The Binary Segmentation failed to narrow changes down to one single change point despite increasing ϵ several times. Since a high ϵ can make the model less accurate and less sensitive to change, it might cause issues with increasing it too much. In Figure 9 the top chart shows Binary Segmentation detected two change points while the bottom chart shows the Bottom-Up approach detecting our change as a single change point. Both of these methods do show a sudden shift of the overall mean can be easily detected and narrowed down to one or two points by CPD.

It was more difficult to narrow it down to a single change point with the gradual drop in the mean in the Weight on Bit data to zero. In Figure 10 we use both the Bottom-Up and Binary Segmentation CPD methods to try to find a single change point in the Weight on Bit. While it would be ideal to observe one or two change points in this dataset and others similar to it, we were unable to adjust ϵ meaningfully to observe results like that.

However, instead of being concerned with the mean shifting to some arbitrarily high number, with Weight on Bit we are concerned

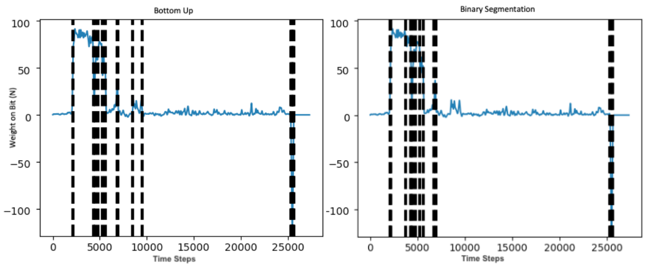

Figure 10: Results of CPD of Weight on Bit using Bottom-Up and Binary Segmentation methods.

when it approaches and hovers around zero in conjunction with sudden increases in our Torque and a sudden decrease in Rate of Penetration. Depending on the specific parameters of any future MDP or other methods, it might be sufficient to analyze change points only in Torque and Rate of Penetration and monitor Weight on Bit to see if it is below a certain threshold.

6 DISCUSSION

Time Series Analysis and Predictions. One of the reasons we selected DBA was its insensitivity to ordering and avoidance of iterative pairwise averaging [18]. However, with DBA in python using tslearn, we noticed that processing two data-sets was fairly quick, but when processing more than three datasets concurrently, the CPU killed the process nearly every iteration. Because Python tends to be very CPU intensive, comparing three data-sets of length i, j, and k made the DTW algorithm run at $O(ijk)$, which assuming the data sets were all of similar length, made our time complexity an inefficient $O(n^3)$. Even with the option to use NVIDIA's Cuda Toolkit [4] with Python, we could improve complexity by writing the DTW algorithm in a language like C++ or C instead of Python and hand-off processes to the GPU more efficiently. This will be explored if we find more datasets outside the average length range.

The evaluation of forecasting models for fault detection in drilling operations has highlighted the advantages of using Long Short-Term Memory (LSTM) networks over traditional machine learning methods like Support Vector Machines (SVM), Random Forest, and AdaBoost. Our analysis, as illustrated in the comparative performance table and figures, underscores the LSTM model's exceptional ability to predict future data points with high accuracy, as evidenced by its highest R-squared value and the lowest MSE.

One insight from our study is the LSTM's handling of the complexities and temporal dependencies inherent in time series data. The LSTM architecture, which emphasizes recent observations through its memory cells, proves to be particularly beneficial for early fault detection. The ability to forecast up to 100 points ahead with considerable accuracy provides a significant lead time, enabling preemptive actions to mitigate or avoid potential faults.

In our exploration of predictive maintenance within drilling operations, a central focus has been on the accurate forecasting of critical operational parameters such as Torque, Weight on Bit, and Rate of Penetration. While these variables provide invaluable insights into the drilling process, our analysis suggests a need to advance beyond traditional forecasting. Specifically, we propose quantifying the likelihood of equipment failure on a scale from 0 to 1 for example, transforming our predictive model into a robust failure probability estimator. This approach not only enhances the interpretability of the forecasts but also enables the setting of threshold values that could signal imminent operational failures.

The contrast in model performances observed in our results section, particularly the higher accuracy of the SVM in forecasting torque from related parameters versus the LSTM's challenges, underscores an insight into predictive modeling for drilling operations. This variance highlights a pivotal consideration: the direct prediction of parameters such as Torque, Weight on Bit, and Rate of Penetration, while useful, may not optimally harness the potential of sophisticated models like LSTM in contexts where the prediction target is closely intertwined with the input features.

Furthermore, this pivot towards failure probability estimation, while enhancing model interpretability, complements the application of CPD in our predictive maintenance framework. CPD, with its capability to detect significant shifts in data patterns, becomes particularly valuable when aligned with the probabilistic outputs of our predictive models. By integrating these probabilistic forecasts with CPD, we can achieve a more dynamic and contextually aware fault detection mechanism. This integration not only capitalizes on the strengths of both methodologies but also addresses the critical need for timely and accurate fault detection in drilling operations.

Change Point Detection. Depending on our data sets, we may have to adjust ϵ to catch slightly smaller significant shifts in the data. This will be determined as we continue to sort through test data and acquire new data. We may also have to adjust the algorithm as change points can differ due to environmental factors in a planetary environment or the composition of the materials drilled into. Despite these considerations, CPD algorithms will be an excellent tool for helping detect indicators of faults early to prevent them from happening, especially as we move to consider online detection methods.

7 CONCLUSIONS AND FUTURE WORK

Our primary goal with this paper was to focus on the analysis of field data and its impacts on future model development. We were able to achieve the goal of evaluating and understanding the fault data from TRIDENT and projecting the data to the same time domain for time series analysis. Through our analysis of the data using forecasting methods, we can conclude the time series data from the TRIDENT drill follow predictable trends when leading up to a fault, and those trends can be detected through offline CPD methods. We believe this will provide a basis for further work including establishing a robust method for data predictions. Additionally, we plan to incorporate situational factors into our methods, including accounting for different planetary sediments and terrain. We will use these findings to create a deterministic model for TRIDENT to understand its rate of progress on a task and the situational awareness of when a situation might escalate to a drilling fault.

ACKNOWLEDGMENTS

This material is based upon work supported by the Minnesota Space Grant Consortium and the National Science Foundation CSGrad4US Fellowship under Grant No. 2240197.

REFERENCES

[1] Ryan Prescott Adams and David J. C. MacKay. 2007. Bayesian Online Changepoint Detection. arXiv:0710.3742 [stat.ML]

[2] Jushan Bai. 1997. Estimating Multiple Breaks One at a Time. *Econometric Theory* 13, 3 (1997), 315–352. https://doi.org/10.1017/S0266466600005831

[3] G. Brat, M. Gheorghiu, D. Giannakopoulou, and C. Pasareanu. 2008. Verification of Plans and Procedures. In *2008 IEEE Aerospace Conference*. IEEE, 1–8. https://doi.org/10.1109/AERO.2008.4526574

[4] NVIDIA Corporation. 2023. Cuda Toolkit Documentation. https://docs.nvidia.com/cuda/. Accessed: 6 December 2023.

[5] Gilles Dowek, César A. Muñoz, and Corina S. Pasareanu. 2007. A Formal Analysis Framework for PLEXIL. In *Workshop at ICAPS*. https://api.semanticscholar.org/CorpusID:16406491

[6] Piotr Fryzlewicz. 2014. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics* 42, 6 (Dec. 2014). https://doi.org/10.1214/14-aos1245

[7] B. Glass. [n. d.]. Atacama Rover Astrobiology Drilling Studies (ARADS) Project.

[8] B. Glass. 2012. Robotics and automation for "ICEBREAKER".

[9] B. Glass. 2022. Repurposing drilling control diagnostics for surface edge detection and boundary advisement during planetary drilling. *ASCE* (2022).

[10] B. Glass, D. Bergman, V. Parro, L. Kobayashi, C. Stoker, R. Quinn, A. Davila, P. Willis, W. Brinckerhoff, K. Warren-Rhodes, M.B. Wilhelm, L. Caceres, J. DiRuggiero, K. Zacny, M. Moreno-Paz, A. Dave, S. Seitz, A. Grubisic, M. Castillo, R. Bonaccorsi, and the ARADS Team. 2023. The Atacama Rover Astrobiology Drilling Studies (ARADS) Project. *Astrobiology* 23 (Dec. 2023), 1245–1258. Issue 12.

[11] B. Glass, C.Fortuin, H. Battah, and I. King. 2024. TRIDENT Drill Validation Testing in Haughton Crater, Devon Island, Canada.

[12] Brian Glass and Sathya Hanagud. 2005. Drilling automation for subsurface planetary exploration.

[13] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2001. An Online Algorithm for Segmenting Time Series. In *Proceedings 2001 IEEE International Conference on Data Mining*. 289–296.

[14] Zhenyu Liu, Zhengtong Zhu, Jing Gao, and Cheng Xu. 2021. Forecast Methods for Time Series Data: A Survey. *IEEE Access* 9 (2021), 91896–91912. https://doi.org/10.1109/ACCESS.2021.3091162

[15] Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, and Michael Fisher. 2019. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *ACM Comput. Surv.* 52, 5, Article 100 (sep 2019), 41 pages. https://doi.org/10.1145/3342355

[16] R.C. Moeller, L.Jandura, and K. Rosette et al. 2021. The Sampling and Caching Subsystem (SCS) for the Scientific Exploration of Jezero Crater by the Mars 2020 Perseverance Rover. *Space Sci Rev* 217, 5 (2021). https://doi.org/10.1007/s11214-020-00783-7

[17] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams. 1998. Remote Agent: to boldly go where no AI system has gone before. *Artificial Intelligence* 103, 1–2 (Aug. 1998), 5–47. https://doi.org/10.1016/S0004-3702(98)00068-X

[18] François Petitjean, Alain Ketterlin, and Pierre Gançarski. 2011. A Global Averaging Method for Dynamic Time Warping, with Applications to Clustering. *Pattern Recogn.* 44, 3 (March 2011), 678–693. https://doi.org/10.1016/j.patcog.2010.09.013

[19] Camilo Rocha, César Munoz, and Héctor Cadavid. 2009. A Graphical Environment for the Semantic Validation of a Plan Execution Language. In *2009 Third IEEE International Conference on Space Mission Challenges for Information Technology*. IEEE, 201–207. https://doi.org/10.1109/SMC-IT.2009.31

[20] H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

[21] Laura Sánchez-García, Miguel Fernandez-Martinez, Mercedes Moreno-Paz, Daniel Carrizo, Miriam García-Villadangos, Juan Manuel Manchado Ortega, Carol Stoker, Brian J. Glass, and Victor Parro. 2019. Multianalytical detection of biomarkers of a 1m-depth drill in an acidic Mars-analog river bed (Rio Tinto): Simulating a future drilling and bioanalysis on Mars. In *The Astrobiology Science Conference (AbSciCon) 2019*. Article 406-9, 406-9 pages.

[22] S. Sarkar, N. Bose, S. Bhattacharya, and S. Bhandari. 2022. The Curious Case of Missing Drill Core: An Investigation on SuperCam Derived IR and LIBS Spectra from the First Drill Site of Mars 2020 Perseverance Rover Mission. In *53rd Lunar and Planetary Science Conference (LPI Contributions, Vol. 2678)*. Article 1966, 1966 pages.

[23] David Schultz and Brijnesh Jain. 2017. Nonsmooth Analysis and Subgradient Methods for Averaging in Dynamic Time Warping Spaces. arXiv:1701.06393 [cs.CV].

[24] Shannon M. Statham. 2011. *Autonomous structural health monitoring technique for interplanetary drilling applications using laser doppler velocimeters*. Ph. D. Dissertation. Georgia Institute of Technology, Atlanta, Georgia.

[25] C.R. Stoker. 2024. Field tests with TRIDENT drill in Bishop tuff help prepares for future missions to Moon and Mars. Universities Space Research Association.

[26] T. Stucky. 2018. Autonomous Regolith Extraction using Real-Time Diagnostics and Dynamic Plan Execution for 1 Meter Class Interplanetary Rotary-Percussive Drills. In *16th Biennial International Conference on Engineering, Science, Construction, and Operations in Challenging Environments*. American Society of Civil Engineers (ASCE), Reston, VA.

[27] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. 2020. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research* 21, 118 (2020), 1–6. http://jmlr.org/papers/v21/20-091.html

[28] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Process.* 167, C (feb 2020), 20 pages. https://doi.org/10.1016/j.sigpro.2019.107299

[29] K. Zacny. 2017. Development and Testing of The Lunar Resource Prospector Drill.